图灵程序设计丛书

写给网络架构师、服务器工程师的

图解服务器端网络架构



467张图表
讲透基础技术和设计要点

硬件构成 · VLAN设计 · IP地址设计
安全设计 · 负载均衡设计 · 冗余配置
虚拟化 · 应用管理

[日] 宫田宽士 著
曾薇薇 译
乌尼日其其格 审

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

版权信息

书名：图解服务器端网络架构

作者：[日]宫田宽士

译者：曾薇薇

ISBN：978-7-115-38817-9

版权声明

译者序

提一个问题：《蒙娜丽莎》为什么如此经典？因为这幅作品不仅凝聚了达·芬奇非凡的绘画技巧，还融合了他对人体解剖学和黄金分割律等诸多理论的理解和应用。不同范畴的技术一经交汇，往往能碰撞出迷人的火花，《蒙娜丽莎》便是个极好的例子。

在茫茫书海中挑出了这本书的你，让我恭喜你一下，如果想要系统地学习服务器端网络架构知识并希望学以致用，这绝对是当前已出版的书籍中最好的一本。不管你是网络架构师、服务器工程师，还是二者兼顾多元并修的 IT 技术牛人，这本书都能让你豁然开朗、如虎添翼！

计算机之间的网络互连始于 20 世纪 80 年代。在我国，互联网从 1997 年才开始得到真正的发展。网络架构本身就是非常年轻的技术，谈到服务器端网络架构这个更为细化的范畴，就更是缺少系统化的指南和探讨了。悄悄地问一下自己，清楚网络设计中的每项设计工程分别是为哪个 OSI 层服务的吗？如果回答是否定的，是不是有点担心设计内容会漏洞百出呢？没关系，悄悄地读透这本秘籍吧，它会让你的功力在不知不觉中大增的。

看到这里，你是不是在担心这本书对自己来说太过高大上，害怕无法驾驭呢？请放心，这本书的门槛并不算高，即便是初学者也能看懂。当你踏踏实实跨过这道门槛后，更加丰富立体的世界就会展现在你的眼前。在如今的 IT 行业中，网络架构师和服务器工程师往往一抓一大把，但能够将二者有机结合并且运用自如的技术人员却并不多见。这本书或许不能让你一口气临摹出有如成品的《蒙娜丽莎》，但起码能让你慢慢勾勒出深得名画精髓的素描图，那是不是也很厉害呢？

总结起来，这本书除了系统化的讲解之外还具有以下两大特色。

（1）细致。拿到原稿时首先让我惊讶的是，居然有一个章节（第 0 章）是专门讲解如何通览到了许多容易忽视但非常重这本书的。学习讲究提纲挈领，这本书细致到了告诉读者应该怎样去提纲挈领，实属难得。书中还提要的设计细节，令人叹服。

（2）实用。入门书容易偏重于理论讲解而缺少实例说明，本书却列举了大量活生生的例子，对实战人员来说具有真正的参考价值，其中不乏让人哈哈大笑的失败事例。而且图文并茂，400 多张辅助说明的图浅显易懂，让你大呼“So easy”。

顺便提一句，原书出自日本软银公司旗下的出版社，软银公司实力如何世人皆知，它旗下的出版社眼力向来不俗，相信读过本书后你也一定会赞同。

我是抱着虚心学习的态度完成这本书的翻译的。虽然从事 IT 行业已近十年，兼做翻译（含口译）也有十来年的丰富经验，但对网络技术和基础架构毕竟未曾做过深入的专攻，以至于对某些内容和文字的考证推敲到了“战战兢兢，如履薄冰”的地步。不过也正因为如此，在翻译的过程中我非常真切地体会到了该书浅显易懂的写作风格和接触到新知识新技能时的无穷乐趣。感谢人民邮电出版社图灵公司让我见识到了不一样的风景，感谢高宇涵老师在编校中付出的辛勤劳动，衷心希望这本书能为读者们开启一扇窗口，让大家看到另一片海阔天空。

曾薇薇

2014 年 10 月于上海

前言

关于本书

本书将为你讲述服务器端网络架构中的基础技术和设计要点。

最近出现了一股回归场内服务（On-premises，也称内部部署，即公司内部运行）的新潮流，颇有与云计算这股时代大潮相抗衡的趋势。将所有的数据都存放在云上是否安全？人们这种不安的心理引发了对场内服务的回归，还滋生了另一种新的潮流——云计算和场内服务的混合体。在这样不断变化的大环境中，笔者认为场内服务仍将继续存在下去。本书就将结合图表，详细说明其中服务器端网络架构的基础技术和设计要点。

最近这几年，在服务器端网络使用的基础通信技术并没有太大的进步。服务器端大多设在绝不允许服务中断的关键任务环境中，新技术很难渗透，也很难植根于这样的环境。但正因如此，服务器端的多余部分才得以剔除，形成了非常精简单纯的风格。网络的基础技术可以说已经成型了，然而在网络上运行的网络设备和服务器的技术仍然踩着现在进行时的节奏在持续不断地爆发性发展，由此出现了虚拟技术和网络存储技术等基于网络的创新技术。如今，它们已在系统中不可或缺。随着这些技术的发展，人们追求的网络形态和网络设计的方式也在时刻发生着变化，基础架构工程师和服务器工程师必须能灵活应对这些变化才行。

在网络世界里，无论出现怎样的新技术，基础部分都不会有太大的变化，无非是在某些地方对某些功能分而化之，或是恰恰相反，将某些地方的某些功能整而合之，使它们周而复始地聚散离合而已。正因为基础技术早已成型，才需要我们更深入、更扎实地掌握它们。只要掌握好基础部分，那么无论上层运行的是什么技术和设备，我们都能够沉着应对，决不会乱了阵脚。

但是，最近重视需求定义和基础设计这些上游工程而轻视详细设计和具体架构这些下游工程的事例有增无减。上游工程当然是重要的，而且“仅了解技术概要即可”这种想法的确会带来更高的效率，有它一定的道理。不过，笔者对这种说法有着明显的抵触感——在下游工程中设置

过多少台机器、经历过多少次问题，最后终究会反映到上游工程中，反映出我们作为基础架构或网络工程师的底蕴和说话的分量。没接触过设备，就不知道哪里才是关键，不知道发生问题时应该如何处理。而且，仅仅将操作手册和规格说明书里的内容囫囵吞枣后就去尝试系统架构，这种网络设计手法也是大错特错的。因为操作手册和规格说明书里毕竟只写了设备所具有的功能，何况在需要绝对稳定的网络世界里，“理解”和“操作”是两个相去甚远的不同概念。现在，不仅存在虚拟环境，又有很多在其中运行的虚拟专用机试用版可供使用，和以往相比，我们可以更方便地进行尝试和验证。所以，在掌握好基础部分之后，还是需要自己去动手设置。在这个过程中经历各种设置和各色问题后，我们才能迈出坚实的下一步。如果本书能够成为众多从事网络工作的工程师们前进的路标，笔者定当备感荣幸。

本书适合的读者

本书适合以下几类读者阅读。

想要设计服务器端的网络工程师

已经掌握了架构和测试等下游工程的工程师会向需求定义和基础设计这些上游工程转移和发展。在网络架构中，基础设计就是生命线，而基础设计中制定的规则决定了服务器端的一切。本书在各章中描述了基础设计中应该确定的最基本的内容，相信能在基础设计中助你一臂之力。

想要了解网络的服务器工程师

如今，虚拟化和网络存储等技术已经成为了系统中不可或缺的部分，这些技术大多采用的是基于网络的框架结构，我们已经可以说服务器和网络是密不可分的。针对那些说着“服务器我很擅长，但网络就不太懂了……”的、稍微有些“偏食”的工程师们，本书使用了大量的图示来讲解，一定会让你喜欢上网络技术的。

负责服务器端运行和管理的现场管理人员

在长期的现场运行中，管理人员会遇到种种问题，例如服务器的服务出现差错、网络设备损坏，等等。排除问题的捷径只有一条，那就是好好学习基础技术。服务器端是一个由诸多基础技术拼接而成的世界，本书列举了一些架构实例，能够帮助你掌握每一项基础技术，最终拼接出一个完整的世界。

谢辞

本书的执笔得力于多方人士的大力协助。软银创新的友保健太先生给了行文迟缓的我极大的关心和鼓励，Gene 先生则带我走进了这个值得为之付出辛勤劳作的写作世界，对于这两位，我感激不尽。在写作过程中我倾注了所有的心力，自己也收获颇丰，因此备感三生有幸。另外还有自己本职工作很忙，却爽快地答应我将旧作《服务器负载均衡入门》一书的创作团队重新组织起来的 Taka 先生、Yosshi 先生，后来加入校对团队的 Nari 先生和 Yuka 女士，以及擅长层面和技能都不相同的各位朋友，正是因为他们严格地把关，才让本书达到了能够在实际工作中派上用场的较高水准。

最后，感谢我的父母和妹妹一家人，他们一直非常宽容地支持我的决定和决心，我在遥远的东京祝愿小外甥悠健康成长。还要感谢红蜻蜓的店主和女掌柜，以后我去造访时也要一如既往地借我电源用哦。

2013 年 12 月　宫田宽士

第 **0** 章　本书的用法

本章概要

本章针对本书的讲解顺序和使用方法进行说明。基础设计是服务器端网络架构中最重要的一个阶段，本书将立足于基础设计的设计细分项目，介绍各细分项目的相关技术和设计要点。基础设计中会制定一些重要规则，这些规则一旦定下便不可动摇，它们对系统今后的可扩展性和可操作性等，乃至对整个网络都将产生重大影响。为了各位读者今后能够进行着眼于未来的基础设计，首先请通过本章了解如何使用本书。

0.1　网络架构的流程

首先说明一下网络架构的大致流程。各位在理解了网络架构的大致流程之后，就可以将该流程与本书的核心——基础设计的设计细分项目和本书内容（技术细分项目和设计细分项目）一一进行匹配。

0.1.1　网络架构分为六个阶段

普通服务器端的网络架构由六个阶段构成，分别为需求定义、基础设计、详细设计、架构、测试和运行。各个阶段并非完全独立，而是前一个阶段为后一个阶段提供输入，彼此之间是密切相关的。

0.1.1.1　需求定义

需求定义是确认客户的需求并将它们一一定义出来的阶段。客户会给出需求方案说明书（RFP，Request For Proposal）来描述他们的需求，但这份资料中只有需求的大致内容。我们应在参考这份资料的基础上听取客户的具体需求，将需求明确和细化，然后整理成一份叫作需求定义书的文档。

0.1.1.2　基础设计

基础设计是确定各要素设置规则的阶段。如果网络设备都属于同一个种类，那么只有设置方法和设置对象不一样而已，它们的功能应该是大同小异的。所以可以这么说，这里定下的规则决定了后面的所有阶段。对网络架构来说，这个阶段是它的生命线。我们应将这里制定的规则整理成一份叫作基础设计书的文档。

0.1.1.3　详细设计

详细设计是根据基础设计中的信息确定各机器参数（设定值）的阶段。毋庸置疑，设置对象会因机器本身、机器种类和 OS 版本等要素的不同而各不相同。我们应详细规定每台机器的设定值，保证无论是谁去看设计书都能够完成设置，进而完成系统架构。我们还应将这里制定的参数整理成一份叫作详细设计书的文档。

不同的企业和供应商对详细设计书有着不同的定义。有的把记载了所有参数的机器设置书和参数表叫作详细设计书，有的则把总结了设定值要点的资料叫作详细设计书，形式多样，不一而足。所以，请事先向客户确认好他们需要什么样的详细设计书。

0.1.1.4　架构

架构是根据详细设计书中的信息（参数、设定值）对机器进行设置的阶段。详细设计书是落实设定值的资料，而在架构这个阶段，我们就要利用这些信息对机器进行设置和连接。毋庸置疑，设置方法也因机器本身、机器种类和 OS 版本等要素的不同而各不相同。我们应根据设置人员的技术水平和客户的要求编写操作手册和操作检查表，将操作步骤细化，力求减少设置失误。

0.1.1.5　测试

测试是指在完成架构的环境中进行单元测试、正常测试、故障（冗余化）测试等各种测试。测试前我们应编写测试规格说明书和测试计划书，然后根据资料中的测试事项进行测试，最后将结果整理成一份叫作测试报告的文档。

在单元测试中应确认例如 LED 能否正常亮灯、OS 能否以预想的版本正常启动、接口能否衔接等事项，以此来确认单机是否能正常运行。

在正常测试中应确认机器在连接的环境中是否按我们设想的那样接通、是否能进行应用通信、是否组成了冗余结构、是否能进行 Syslog、SNMP、NTP 等管理通信等事项，以此来确认各相关机器能否彼此协作运行。

在故障测试中应针对所有进行了冗余配置的设备确认如下事项：当该处发生故障时能否继续提供服务，出现故障时会发送怎样的日志，等等。

0.1.1.6　运行

运行是为了持续稳定地提供完成架构的系统而存在的阶段。并不是架构完系统就万事大吉了，而从这里才刚刚开始起步，以后可能有些机件会损坏，也可能会需要新增服务器。所以，除了日常操作之外我们还应多方考虑，做到有备无患。我们应将所有和运行相关的业务整理成一份叫

作运行手册的文档。

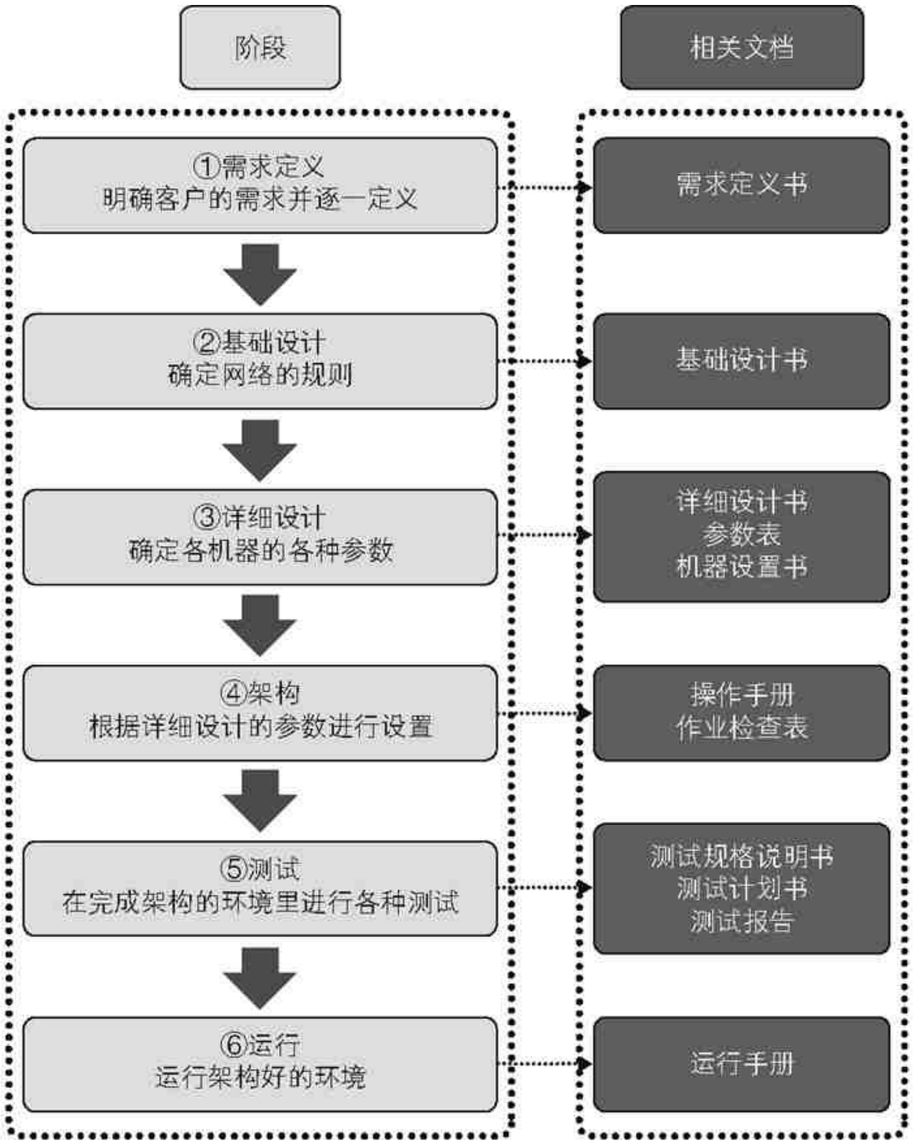


图 **0.1.1**　网络架构分为六个阶段

0.1.2　网络架构的重点是基础设计

网络架构中最重要的阶段就是基础设计阶段。基础设计中确定的规则对系统以后的可扩展性和可操作性等，乃至对整个网络都将产生重大影响。由于不同的客户对基础设计的程度会有不同的要求，我们应在一开始就向客户确认他们所需要的程度，然后根据他们的具体要求进行基础设计。如果有其他的现有系统存在，不妨去察看一下该系统的基础设计

书，了解设计书大致处于怎样的水平。

关于服务器端的网络架构，本书将重点介绍基础设计这一最重要的阶段，并对各设计细分项目所需的技术和设计要点进行说明。普通服务器端的网络架构基础设计可大致分为物理设计、逻辑设计、安全设计与负载均衡设计、高可用性设计、管理设计这五个设计细分项目。当然这些只是一般的例子，设计细分项目会因客户需求、服务性质等的不同而不同。我们可根据实际需要自行调整，对这些细分项目进行切分、删除和添加。

接下来要对各设计细分项目的概要以及本书中与各设计细分项目相关的技术细分项目和设计细分项目进行梳理和说明。希望尝试物理设计的读者请参看第 1 章，希望尝试逻辑设计的读者请参看第 2 章。

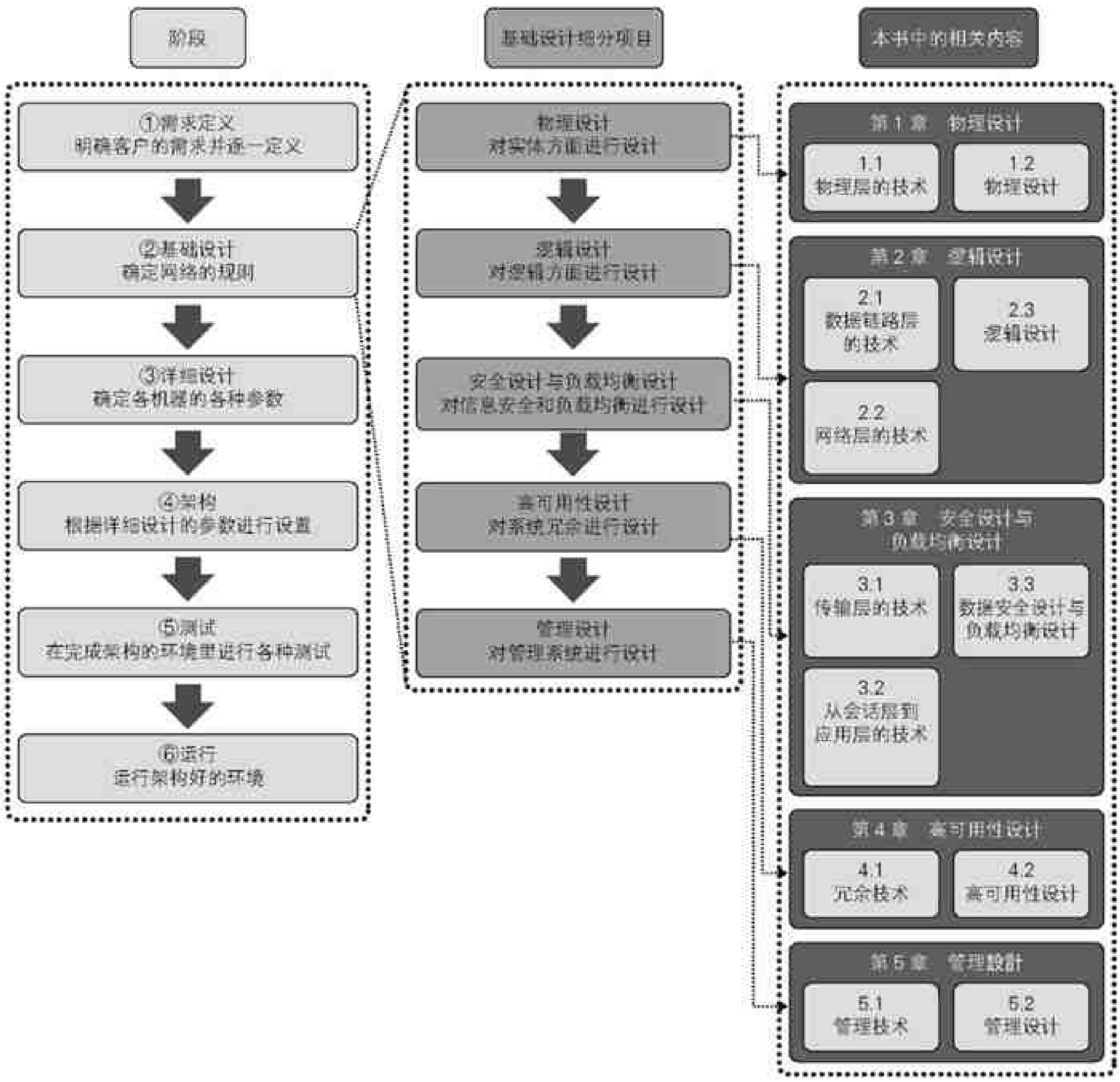


图 **0.1.2**　本书的结构

0.1.2.1　物理设计

在物理设计中，我们要定义服务器端所有实体对象的所有规则。实体对象种类繁多，从线缆到机架、电源，这些都是实体对象。我们要对所有的实体对象都进行严格的定义。

本书的第 1 章讲述物理设计，这章将对物理设计所必需的技术和设计要点进行说明。关于设计细分项目以及本书中与之相关的技术细分项目和设计细分项目的详细内容请参看下表。

表 **0.1.1**　本书中与物理设计相关的技术细分项目和设计细分项目

设计细分项目		设计概要	本书中的相关技术细分项目		本书中的相关设计细分项目	
1	物理设计	对实体方面进行设计	1.1	物理层的技术	1.2	物理设计
1.1	物理设计规则	定义物理设计的整体规则	—	—	—	—
1.2	物理结构设计	如何连接各台机器	—	—	1.2.1	服务器端有两种结构类型
1.3	硬件构成设计	使用什么机器、哪种类别	—	—	1.2.2	选用设备时应参考考查项的最大值
1.4	软件构成设计	使用什么版本的 OS	—	—	1.2.3	选择稳定可靠的 OS 版本
1.5	连接设计	在何处、如何连接	—	—	—	—
1.5.1	线缆设计	在何处、使用什么线缆	1.1.1	物理层里有多种规格	1.2.4	根据实际配置和使用目的选择线缆
1.5.2	物理连接设计	如何设置速率与双工			1.2.5	端口的物理设计出乎意料地重要
1.5.3	端口分配设计	按怎样的顺序使用端口	—	—		
1.6	设备设计	如何使用机架和电源	—	—	—	—
1.6.1	机架安装设计	如何将机器安装到机架上	—	—	1.2.6	巧妙地配置设备
1.6.2	电源连接设计	如何连接电源	—	—		

0.1.2.2　逻辑设计

在逻辑设计中，我们要定义服务器端所有逻辑对象的所有规则。任何网络都是立足于物理上的逻辑成立的，我们应对如何划分 VLAN、如何分配 IP 地址等规则一一做出定义。

本书的第 2 章讲述逻辑设计，这章将对逻辑设计所必需的技术和设计要点进行说明。关于设计细分项目以及本书中与之相关的技术细分项目和设计细分项目的详细内容请参看下表。

表 **0.1.2**　本书中与逻辑设计相关的技术细分项目和设计细分项目

设计细分项目		设计概要	本书中的相关技术细分项目		本书中的相关设计细分项目	
2	逻辑设计	进行逻辑方面的设计	2.1	数据链路层的技术	2.3	逻辑设计
			2.2	网络层的技术		
2.1	逻辑设计规则	定义逻辑设计的整体规则	—	—	—	—
2.2	VLAN 设计	如何划分 VLAN	2.1.1	数据链路层是物理层的帮手	2.3.1	整理出所需的 VLAN
			2.1.2	数据链路层的关键在于 L2 交换机的运作		
			2.1.3	ARP 将逻辑和物理关联到一起		
2.3	IP 地址设计	如何分配 IP 地址	2.2.1	网络是由网络层拼接起来的	2.3.2	在考虑数量增减的基础上分配 IP 地址
			2.2.4	自动设置 IP 地址的 DHCP		
2.4	路由设计	如何进行路径选择	2.2.2	将网段连接起来	2.3.3	路径选择以简为上
			2.2.5	用于故障排除的 ICMP		
2.5	地址转换设计	如何转换地址	2.2.3	转换 IP 地址	2.3.4	NAT 要按入站和出站分别考虑

0.1.2.3　安全设计与负载均衡设计

在安全设计中，我们要定义防火墙的基本规则。当今，信息安全的重要性是无需赘言的。在安全设计中制定明白易懂、简单实用的安全规则可以说是信息安全的基础。此外，我们还要在负载均衡设计中定义服务器的负载均衡规则。近来，惊呼信息量加速猛增的声音不绝于耳，在这样的背景下，服务器负载均衡技术有了惊人的发展和进步。可以这么认为，在所有的大规模服务器端，服务器负载均衡技术都默默无闻地奉献着一己之力。在负载均衡设计中，高效的信息量负载均衡要承担两个重要职责，那就是保障服务器端的可扩展性和灵活性。

本书的第 3 章讲述安全设计与负载均衡设计，这章将对安全设计与负载均衡设计所必需的技术和设计要点进行说明。本来将它们的设计细分项目分开讲述也是可以的，不过因为二者都与从传输层到应用层的技术相关，就将它们合并在一起解说了，各位读者可根据需要将它们的细分项目分开阅读。关于设计细分项目和本书中与之相关的技术细分项目以及设计细分项目的详细内容请参看下表。

表 **0.1.3**　本书中与安全设计、负载均衡设计相关的技术细分项目和设

计细分项目

设计细分项目		设计概要	本书中的相关技术细分项目		本书中的相关设计细分项目	
3	数据安全设计与负载均衡设计	进行与安全和负载均衡相关的设计	3.1	传输层的技术	3.3	数据安全设计与负载均衡设计
			3.2	从会话层到应用层的技术		
3.1	安全设计规则	定义安全设计的整体规则	—	—	—	—
3.2	安全设计	如何确保安全性	—	—	—	—
3.2.1	通信需求的整理	将发生怎样的通信	3.1.1	通过端口号划分服务器进程	—	—
			3.2.1	HTTP 支撑着互联网		
			3.2.2	用 SSL 保护数据		
			3.2.3	用 FTP 传输文件		
			3.2.4	用 DNS 解析名称		
3.2.2	对象设计	如何定义网络对象和协议对象	3.1.2	用防火墙守卫系统	3.3.1	数据安全设计
3.2.3	安全区域设计	如何分配安全区域				
3.2.4	安全基本规则的设计	如何定义安全基本规则				
3.3	负载均衡的设计规则	定义负载均衡设计的整体规则	—	—	—	—
3.4	负载均衡设计	如何进行负载均衡	—	—	—	—
3.4.1	负载均衡需求的整理	应针对什么通信进行负载均衡	3.1.3	通过负载均衡器分散服务器的负荷	3.3.2	负载均衡设计
3.4.2	健康检查设计	使用什么健康检查				
3.4.3	负载均衡方式的设计	使用什么负载均衡方式				
3.4.4	持久化设计	使用什么持久化方式				
3.4.5	SSL 加速设计	是否使用 SSL 加速				

0.1.2.4 高可用性设计

在高可用性设计中，我们要定义系统冗余的相关规则。服务器端大多设在关键任务环境中，不允许出现服务中断的情况，因此我们要对所有部分进行冗余设计，确保无论在哪里发生怎样的故障都能够从容应对。

本书的第 4 章讲述高可用性设计，这章将对高可用性设计所必需的技术和设计要点进行说明。关于设计细分项目以及本书中与之相关的技术细分项目和设计细分项目的详细内容请参看下表。

表 **0.1.4**　本书中与高可用性设计相关的技术细分项目和设计细分项目

设计细分项目		设计概要	本书中的相关技术细分项目		本书中的相关设计细分项目	
4	高可用性设计	进行高可用性相关的设计	4.1	冗余技术	4.2	高可用性设计
4.1	高可用性设计规则	定义高可用性设计的整体规则	—	—	—	—
4.2	链接冗余设计	使用哪种链接冗余方式	4.1.1	物理层的冗余技术	4.2.1	高可用性设计
4.3	机箱冗余设计	使用哪种机箱冗余方式				
4.4	STP 设计	使用哪种 STP 模式、使用哪种交换机进行路由桥接	4.1.2	数据链路层的冗余技术		
4.5	FHRP 设计	使用哪种 FHRP、激活哪个 L3 交换机	4.1.3	网络层的冗余技术		
4.6	路由协议冗余设计	使用哪种路由协议实现冗余				
4.7	防火墙冗余设计	如何实现防火墙冗余	4.1.4	从传输层到应用层的冗余技术		
4.8	负载均衡器的冗余设计	如何实现负载均衡器冗余				
4.9	通信流设计	将会发生怎样的通信流	—	—		
4.9.1	正常情况时的通信流	正常情况时经过怎样的路由	—	—	4.2.2	理清通信流
4.9.2	异常情况时的通信流	异常情况时经过怎样的路由	—	—		

0.1.2.5　管理设计

在管理设计中，我们要定义与服务器端运行管理相关的所有规则。管理设计中的严格规定直接影响后面运行阶段的可操作性和可扩展性。

本书的第 5 章讲述管理设计，这章将对管理设计所必需的技术和设计要点进行说明。关于设计细分项目以及本书中与之相关的技术细分项目和设计细分项目的详细内容请参看下表。

表 **0.1.5**　本书中与管理设计相关的技术细分项目和设计细分项目

设计细分项目				设计概要	本书中的相关技术细分项目		本书中的相关设计细分项目	
5	管理设计			进行运行管理相关的设计	5.1	管理技术	5.2	管理设计
	5.1	管理设计规则		定义管理设计的整体规则	—	—	—	—
	5.2	主机名设计		定义怎样的主机名	—	—	5.2.1	确定主机名
	5.3	标签设计		粘贴怎样的标签	—	—	5.2.2	通过标签管理连接
		5.3.1	线缆标签设计	在线缆上粘贴怎样的标签	—	—		
		5.3.2	机体标签设计	在机体上粘贴怎样的标签	—	—		
	5.4	密码设计		定义怎样的密码	—	—	5.2.3	设计密码
	5.5	备份与修复设计		如何备份设置、如何进行修复	—	—	5.2.4	管理设置信息
	5.6	同步设计		在何处进行时间同步	5.1.1	用 NTP 同步时间	—	—
	5.7	SNMP 设计		在何处定义 SNMP	5.1.2	用 SNMP 探测故障	—	—
	5.8	Syslog 设计		在何处定义 Syslog 服务器、使用什么程序模块和严重性	5.1.3	用 Syslog 探测故障	—	—
	5.9	CDP/LLDP 设计		在何处激活 CDP/LLDP	5.1.4	传递设备信息	—	—

第 1 章　物理设计

本章概要

本章主要介绍用于服务器端的物理层技术、使用该技术时的设计要点以及一般的物理结构类型。

在服务器端，我们能看到的物体只有机架、线缆、端口等具有物理性质的东西。正因如此，我们要充分理解它们的技术和规格，做出符合客户需求的设计。物理层中的严格设计会深深影响系统今后的可扩展性和运行管理性。

1.1　物理层的技术

物理层是所有网络技术的根基。如今，随着网络的高速发展，存储数据、音频数据等各种各样的数据都已经能够在网络中流通。所有的这些数据信息都需要经过物理层才能流通，如果没有物理层，网络就无法连通，我们就不能收发邮件，也不能连接互联网。

1.1.1　物理层里有多种规格

正如其名，物理层是承担着通信任务的所有物理实体所在的层，位于 OSI 参考模型的最下层。它的名字和其他层比起来似乎有点一本正经，不过不用想得太复杂。在公司和学校我们常 常能看到 LAN 网线，把 LAN 网线理解成物理层就行了。如果用的是无线 LAN，那就把信号理解成物理层好了。

计算机的世界是由 0 和 1 这两个数字构成的。0 和 1 叫作比特，连续的比特叫作比特流。数据链路层向物理层发送由 0 和 1 构成的数据（帧），物理层接收后将它们转换为电信号和光信号。物理层中规定了数据在传输媒介（线缆）中的流动规则，还对所有物理性质的要素做了规定，如线缆的材质、连接器的形状、PIN 的接法等。

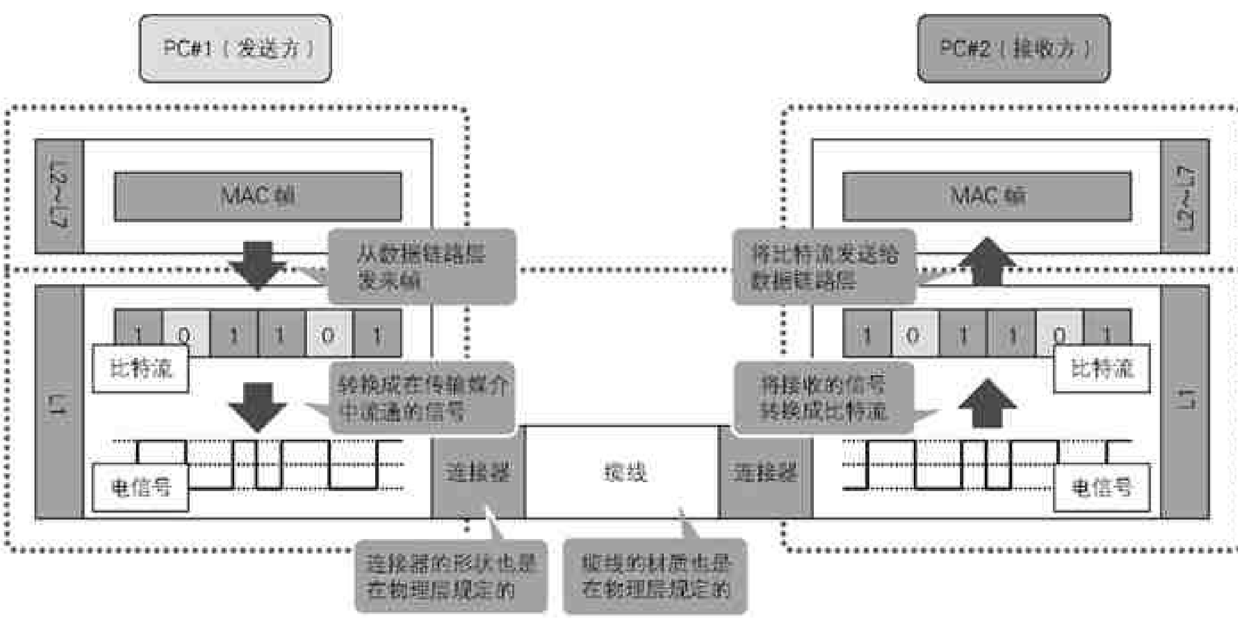


图 **1.1.1**　通过物理层数据在传输媒介中流动

1.1.1.1　规格整理好后物理层就会水落石出

在讲解物理层的要点时会出现大量的规格名称。因此，我们先来整理一下规格。规格整理好了，物理层也就整理好了，大家也就能够理解了。

物理层的标准化并不是在自身范围内完成的，而是和数据链路层在一起成套实现的，整理规格时要将这点考虑进去。有一个叫作 IEEE802 委员会的国际标准化组织对物理层和数据链路层的相关技术标准化起到了推动和促进作用。IEEE802 委员会下设很多工作组（WG，Working Group），负责探讨不同的专业领域，人们以委员会名后面加上小数点，即类似 IEEE802.1 这样的命名形式来区分它们。而工作组制定出的标准则会进一步在后面加上英文字母，即以 IEEE802.1x 这样的形式来命名。工作组数量极多，有的已经停止活动，有的甚至早已解散，所以我们没有必要去掌握所有的规格。在理解了物理层之后，我们只要掌握两个关键标准就足够了，一个是以太网（Ethernet）标准 IEEE802.3，另一个则是无线 LAN 标准 IEEE802.11。只要掌握好它们，基本上就可以应对所有的网络了。本书仅涉及服务器端需要用到的以太网。

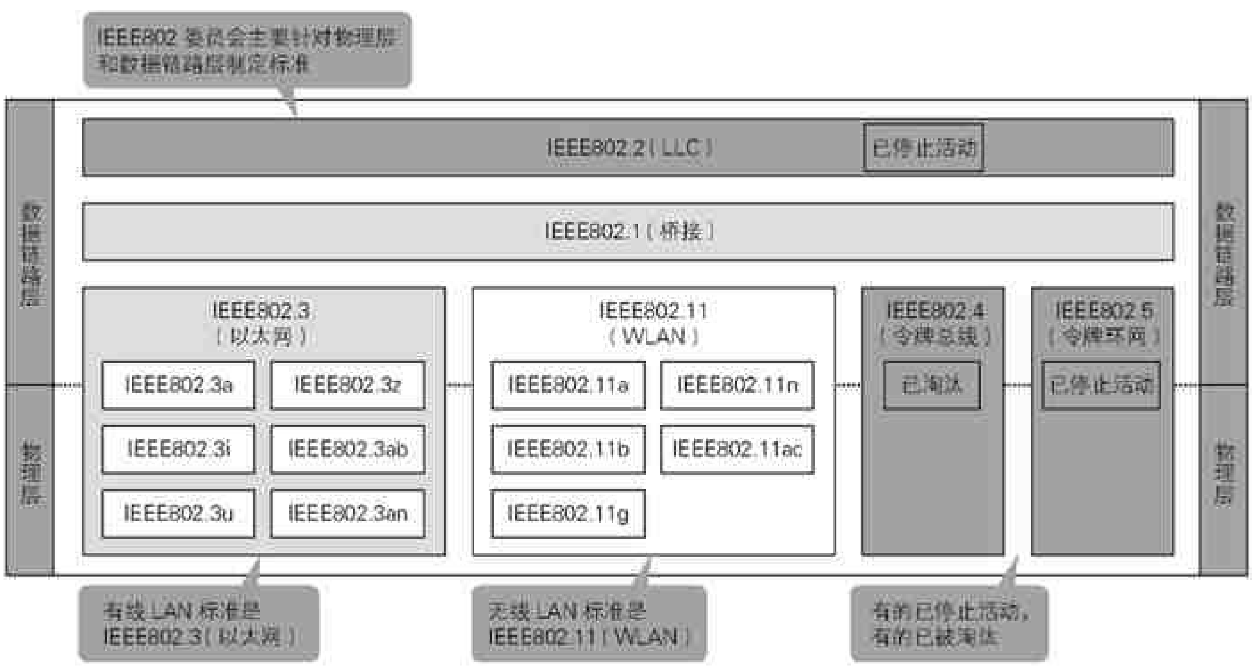


图 **1.1.2**　要掌握 **IEEE802.3** 和 **IEEE802.11**

以太网标准另有称呼

以太网是处理物理层和数据链路层的一种标准。在现代网络的事实标准

中，可以说目前大部分有线 LAN 环境都是以太网。IEEE802 委员会规定用 IEEE802.3 对以太网进行标准化管理。

IEEE802.3 中有多种标准，都是在 IEEE802.3 后面加上不同的英文字母命名的。然而在以太网中这些标准名称几乎不怎么用，人们一般用表示这些标准概要的别名来称呼它们。例如，使用双绞线电缆的千兆位以太网标准是 IEEE802.3ab，别名叫作 1000BASE-T。大多数场合人们都不用 IEEE802.3ab 这个名字而是称呼它为 1000BASE-T。下表中列出了各种标准的名称和它们所对应的别名，一般来说，看别名就能大致了解这是怎样的规格。

表 **1.1.1**　人们常常用别名来称呼以太网

BASE 后面如果是英文字母，表示传输媒介或激光的种类。
T：双绞线电缆
S：短波长激光
L：长波长激光

传输速度	传输媒介			激光的种类	
	同轴电缆		双绞线电缆	短波长激光	长波长激光
10Mbit/s	10BASE5 / IEEE802.3	10BASE2 / IEEE802.3a	10BASE-T / IEEE802.3i		
100Mbit/s			100BASE-TX / IEEE802.3u		
1Gbit/s			1000BASE-T / IEEE802.3ab	1000BASE-SX / IEEE802.3z	1000BASE-LX / IEEE802.3z
10Gbit/s			10GBASE-T / IEEE802.3an	10GBASE-SR / IEEE802.3ae	10GBASE-LR / IEEE802.3ae

BASE 后面接数字时表示距离

BASE 前面的数字表示传输速度
10　10Mbit/s
100　100Mbit/s
1000　1Gbit/s
10G　10Gbit/s

由于网络的急速发展，物理层里存在着大量纷杂的通信媒介和标准。光是讲解传输编码和信号转换方式的内容就足够出一本书了，足见其高深。不过，设计和架构有线 LAN 的时候只需要注意两点，那就是线缆和连接器。下面会针对这两个要点，结合它们各自对应的标准进行详细说明。

在物理层一般使用双绞线电缆和光纤光缆这两种线缆。线缆不同，使用的连接器也不同，因此本书会在介绍线缆时一并说明连接器的相关内容。

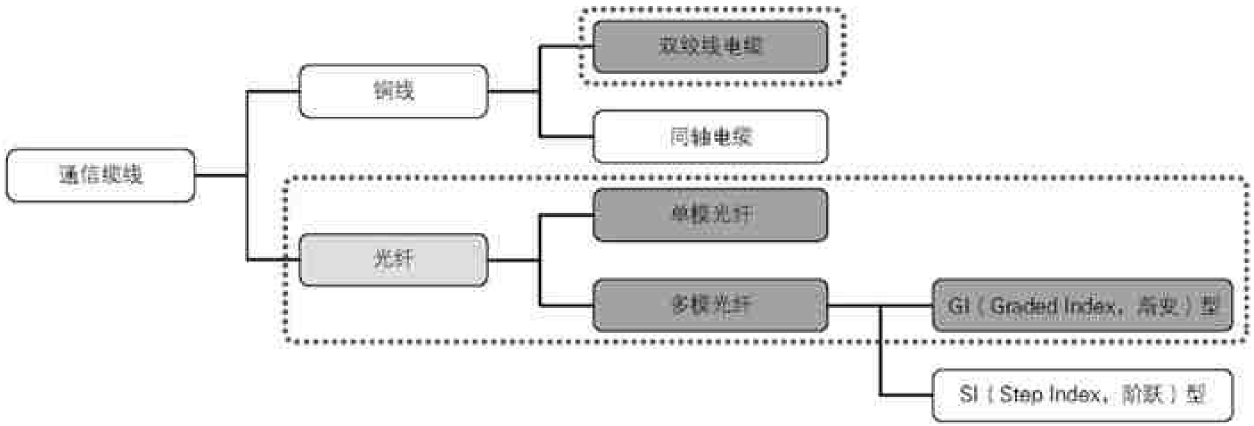


图 **1.1.3**　用于有线 **LAN** 的两种常见线缆

表 **1.1.2**　双绞线电缆和光纤光缆的比较

比较事项	双绞线电缆	光纤光缆
传输媒介的材质	铜	玻璃
传输速度	慢	快
信号的衰减	大	小
受到电磁的干扰	大	无
使用	方便	困难
成本	低	高

1.1.1.2　双绞线电缆有两大要素——类和传输距离

首先我们来看双绞线电缆。双绞线电缆采用○○BASE-T 或○○BASE-TX 这样的标准。○○BASE-T 的 T 是双绞线电缆的英文 Twisted-Pair Cable 的第一个字母。

双绞线电缆看起来是一根电缆，但实际上是八根铜线，每两根（双）相互缠绕（绞），最后再全部绞合成一整根线缆。双绞线电缆分两种，一种是线缆部分用铝箔等材质做了绝缘处理的 STP（Shielded Twist Pair，屏蔽双绞线）电缆，另一种是没做绝缘处理的 UTP（Unshielded Twist Pair，非屏蔽双绞线）电缆。

常见的 **LAN** 网线是 **UTP** 电缆

UTP 电缆就是我们常说的 LAN 网线，在公司、家庭和家电商场等地方经常能看到它们的身影，应该是我们最为熟识的线缆。UTP 电缆因为使用方便、价格便宜，获得了爆发性的普及。最近颜色也变得多种多样了，而且更加纤细，外形也华丽了许多。但是这种电缆不抗电磁干扰，所以不适合在工厂等电磁干扰较多的环境中使用。

STP 电缆则克服了不抗电磁干扰这个缺点。这种电缆做了绝缘处理，有效地减轻了内外部电磁干扰，然而美中不足的是绝缘处理使得电缆用起来不太方便，因此，目前只有在工厂等条件恶劣的环境中人们才会用到它，服务器端基本上并无它的用武之地。

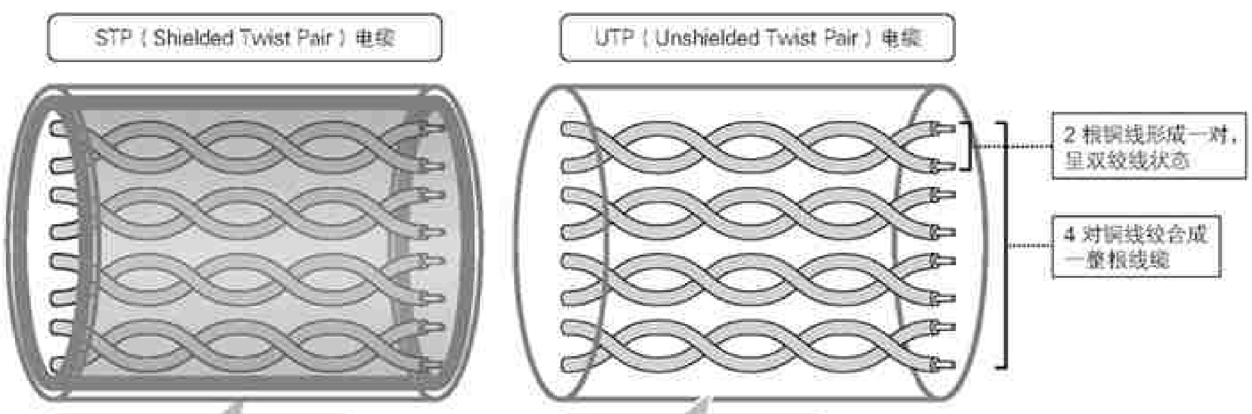


图 **1.1.4　UTP** 和 **STP** 的差异在于是否做了绝缘处理

类的数字越大传输速度就越快

在双绞线电缆中有一个“类”的概念。仔细观察家电商店里卖的 LAN 网线规格表就会发现，上面印着 6 类、5 类这样的标记。这个类代表传输速度，类的数字越大传输速度就越大。

目前以太网环境使用的类都不低于 5，1 到 4 类的线缆无法支持 100BASE-TX、1000BASE-T 这样比较新的标准。因此，如果你想要迁移到目前主流的高速以太网环境，同时还要沿用旧式的线缆环境，就需要注意确认各种线缆对应的规格。

当铜线根数（铜线芯数）较少时尤其需要注意。例如，3 类线缆只有 4 根铜线芯，而 1000BASE-T 需要用到 8 根铜线芯以提高吞吐率。所以在沿用 3 类线缆环境的前提下向 1000BASE-T 环境迁移的话，连对接都完成不了，在架构现场就会窘态百出。我们必须在相应的时机将整个线缆环境全部替换掉才行。

各类电缆及其对应的规格请参看下表。一定要确认无误之后再去选择使用什么样的双绞线电缆。

表 **1.1.3**　类的数字越大传输速度就越快

类	种类	铜线芯数	对应的频率	主要对应的规格	最高传输速度	最大传输距离
3 类	UTP/STP	4 芯2对	16Mbit/s	10BASE-T	16Mbit/s	100m
4 类	UTP/STP	4 芯2对	20MHz	令牌环网	20Mbit/s	100m
5 类	UTP/STP	8 芯4对	100MHz	100BASE-TX	100Mbit/s	100m
5e 类	UTP/STP	8 芯4对	100MHz	100BASE-T	1Gbit/s	100m
6 类	UTP/STP	8 芯4对	250MHz	1000BASE-T 10GBASE-T	1Gbit/s 10Gbit/s	100m 55m（10GBASE-T 时）

6A类	UTP/STP	8 芯4对	500MHz	10GBASE-T	10Gbit/s	100m
7 类	STP	8 芯4对	600MHz	10GBASE-T	10Gbit/s	100m

根据不同情况选择使用直通线或交叉线

在双绞线电缆中，除了类之外还有直通线和交叉线这两个概念。它们的外表可以说毫无二致，不同之处在于从名为 RJ-45 的连接器部位隐约能看到的铜线排列。

直通线和交叉线与名为 MDI 和 MDI-X 的两种物理端口类型有着密切的关系。前面已经介绍过，双绞线电缆是将八根铜线按每两根（双）相互缠绕（绞）的方式绞合，最后再全部绞合成一整根线缆的。服务器、PC 的网卡和交换机的物理端口中有 8 个 PIN，用于接收八根铜线，从左至右都有相应的编号，而且各有各的作用。

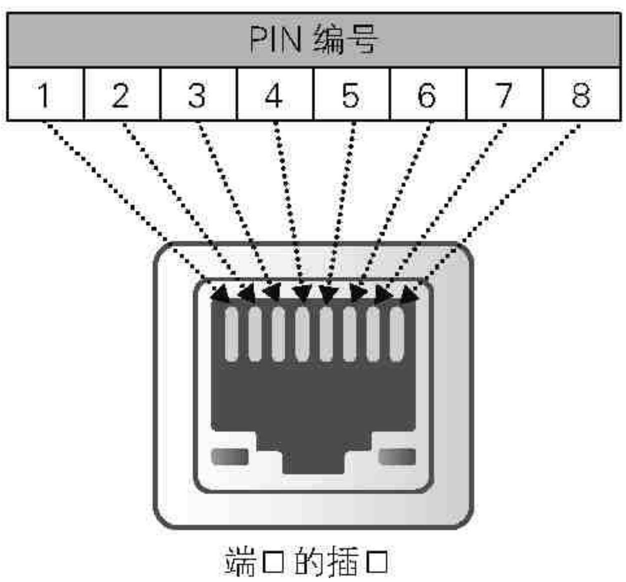


图 **1.1.5**　物理端口中有 **8** 个 **PIN**

在 100BASE-TX 中，MDI 端口通过 1 号和 2 号 PIN 发送信息，通过 3 号和 6 号 PIN 接收信息，剩余的 PIN（4 号、5 号、7 号、8 号）并不使用。PC 和服务器的网卡，还有路由器的物理端口都是 MDI 端口。MDI-

X 端口则和 MDI 端口恰恰相反，通过 1 号和 2 号 PIN 接收信息，通过 3 号和 6 号 PIN 发送信息。L2 交换机和 L3 交换机的物理端口是 MDI-X 端口。

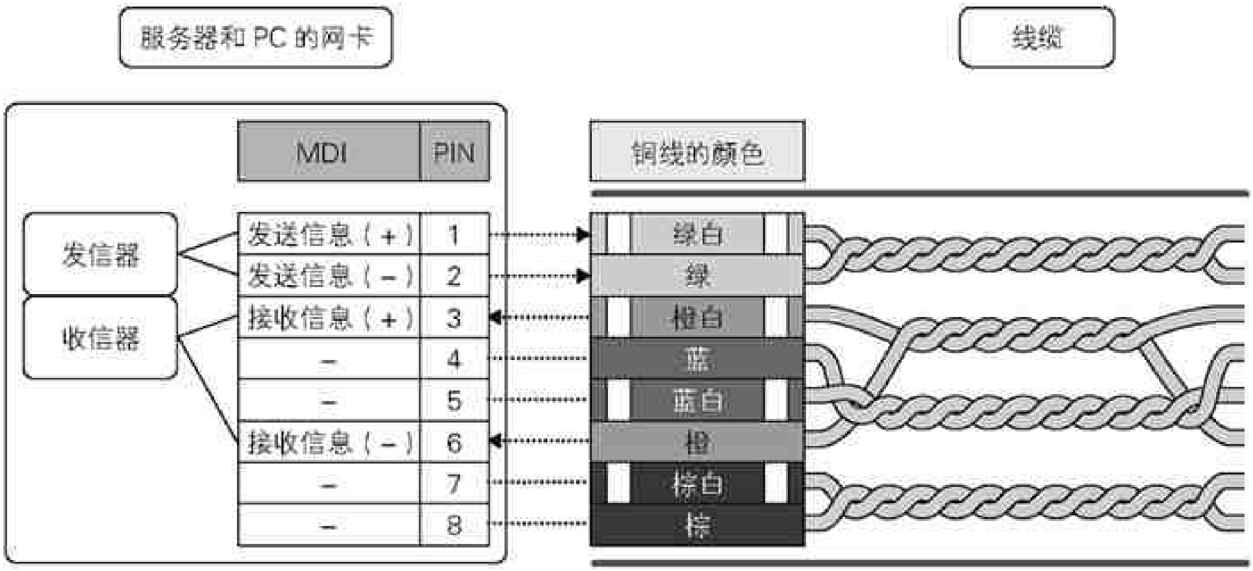


图 **1.1.6　PC** 和服务器的网卡是 **MDI** 端口

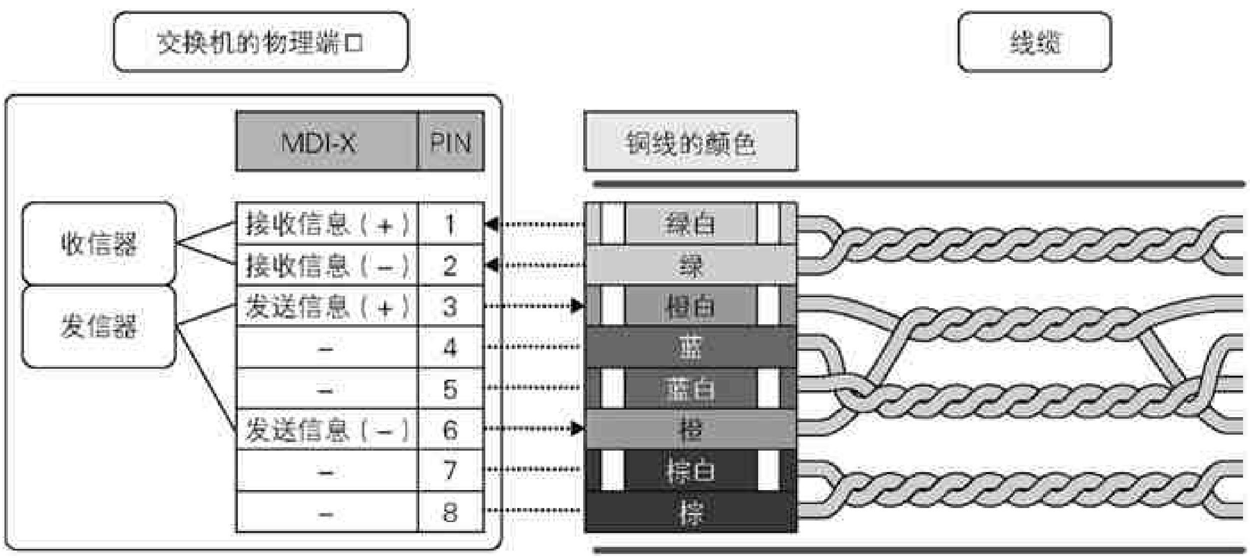


图 **1.1.7**　交换机的物理端口是 **MDI-X** 端口

数据传递时，一方发送了信息，另一方必然就会接收信息。如果双方都要接收信息，就会变成对峙的状态；而如果双方都要发送信息，数据就会发生冲突。因此，当双绞线电缆中的铜线是平行排列的情况时，就必须将 MDI 与 MDI-X 连接起来。这样，MDI 发送信息则 MDI-X 接收信

息，MDI-X 发送信息则 MDI 接收信息。这种铜线为平行排列状态的线缆叫作直通线。举个例子，当 PC 或服务器连接到交换机上时，MDI 和 MDI-X 的上述关系成立，所以可以使用直通线。

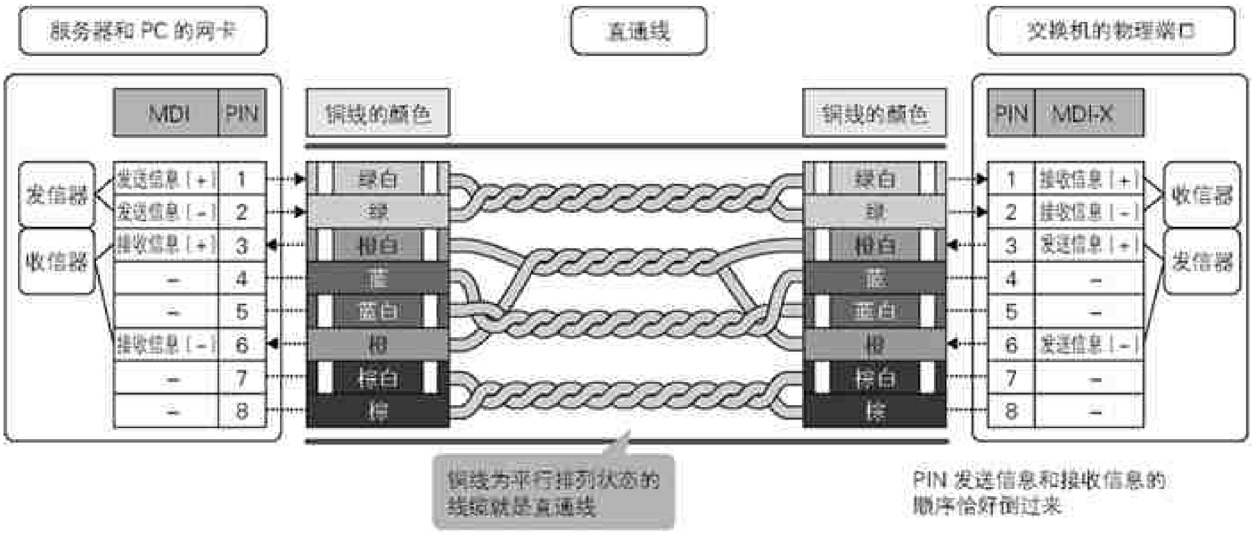


图 **1.1.8　MDI** 和 **MDI-X** 使用直通线连接

然而，MDI 和 MDI-X 的上述关系并非在所有连接环境中都是成立的。有计算机之间互联的情况，也有交换机之间互联的情况。这些情况下，收发信息的关系并不成立，连对接都无法实现。

那么该怎么办呢？遇到这种情况，把布线改一改就好了。这时候我们需要用到的就是交叉线，交叉线的铜线配置在内部恰好是相互颠倒的，这样即使端口类型相同也能形成收发信息的关系。于是，计算机之间的互联也好，交换机之间的互联也好，就都能形成彼此收发信息的关系了。

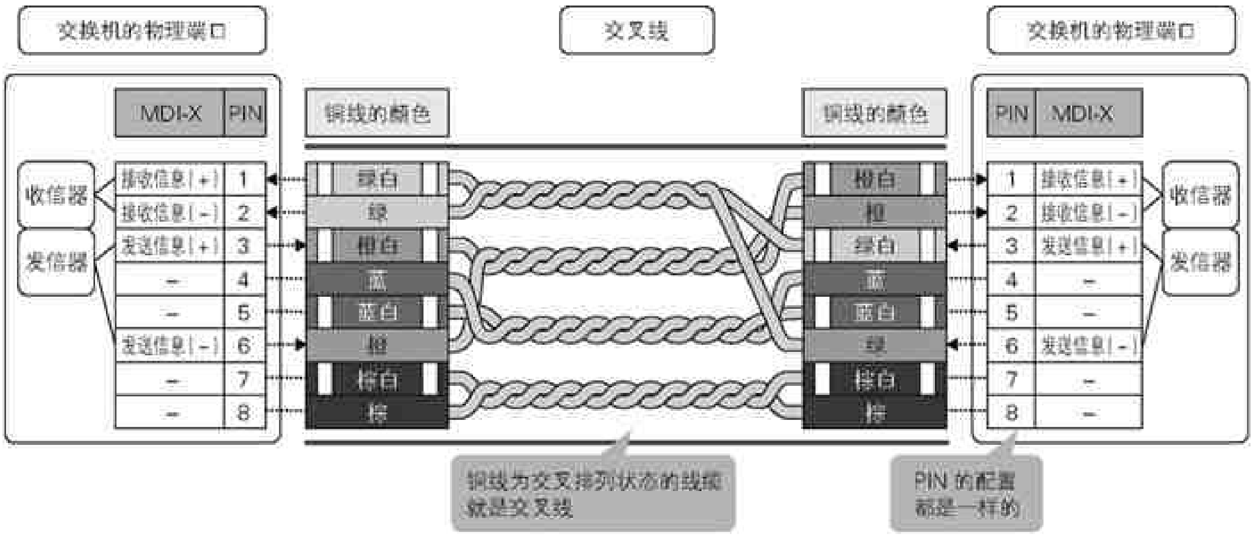


图 **1.1.9**　端口类型相同时应使用交叉线连接

上面介绍了直通线和交叉线，但实际上人们最近不太使用交叉线了，这是为什么呢？以前人们对直通线和交叉线做了严格的使用区分，可是这种严格的使用区分却带来了意想不到的麻烦，慢慢地反而变成了问题的根源。这里要用交叉线、那里要用直通线……像这样指定在哪里用什么线是非常麻烦的，在架构大规模的网络环境时也的确很费事。在这样的背景下，出现了具有能够自动识别端口类型功能（Auto MDI/MDI-X）的交换机。Auto MDI/MDI-X 功能能够自动识别对方的端口类型，并根据该类型将收信器和发信器进行调换。有了这项功能，同样的端口类型也能够使用直通线，就无需用到交叉线了。最近的大多数机器都具备 Auto MDI/MDI-X 功能，于是人们就不再需要每次都考虑线缆的种类了。

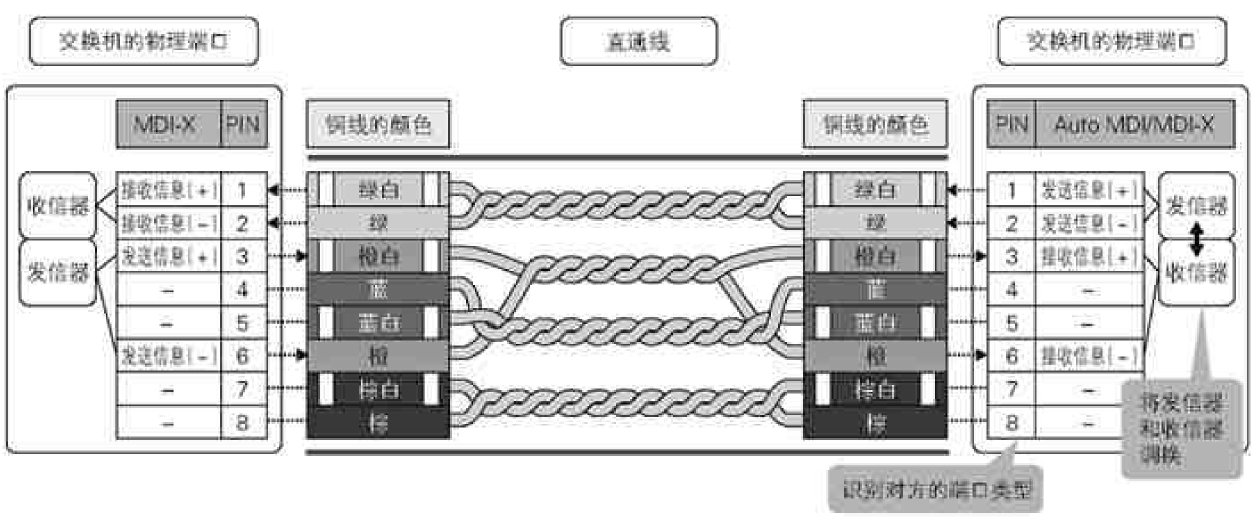


图 **1.1.10**　使用 **Auto MDI/MDI-X** 功能的话任何情况都能用直通线连接

在 **1000BASE-T** 中八根铜线都会用到

在 100BASE-TX 中，MDI 端口通过 1 号和 2 号 PIN 发送信息，通过 3 号和 6 号 PIN 接收信息，剩余的 PIN（4 号、5 号、7 号、8 号）并不使用。这时候，双绞线电缆里虽然有八根铜线却只能用到一半，剩下的四个 PIN 用不上，非常浪费。而在 1000BASE-T 中，剩余的四个 PIN 也能用在数据通信上以提高吞吐率，它是千兆位以太网环境中最常用的标准。IEEE802 委员会赋予 1000BASE-T 的名字为 IEEE802.3ab。

1000BASE-T 的通信并不像 100BASE-TX 那样将收发信分开处理，而是用同样的 PIN 进行数据收发。收发器和 PIN 之间有一个混合电路板，在那里将发送的数据和接收的数据分开，然后分别交给发信器和收信器。在 100BASE-TX 中 1 号和 2 号 PIN 是用于发信的，而在 1000BASE-T 中，1 号和 2 号 PIN 既能用于收信也能用于发信。1 号和 2 号 PIN 进行收发、3 号和 6 号 PIN 进行收发、4 号和 5 号 PIN 进行收发、7 号和 8 号 PIN 进行收发——就像这样，四对 PIN 都能进行数据收发，由此实现了 1Gbit/s 的高速通信。

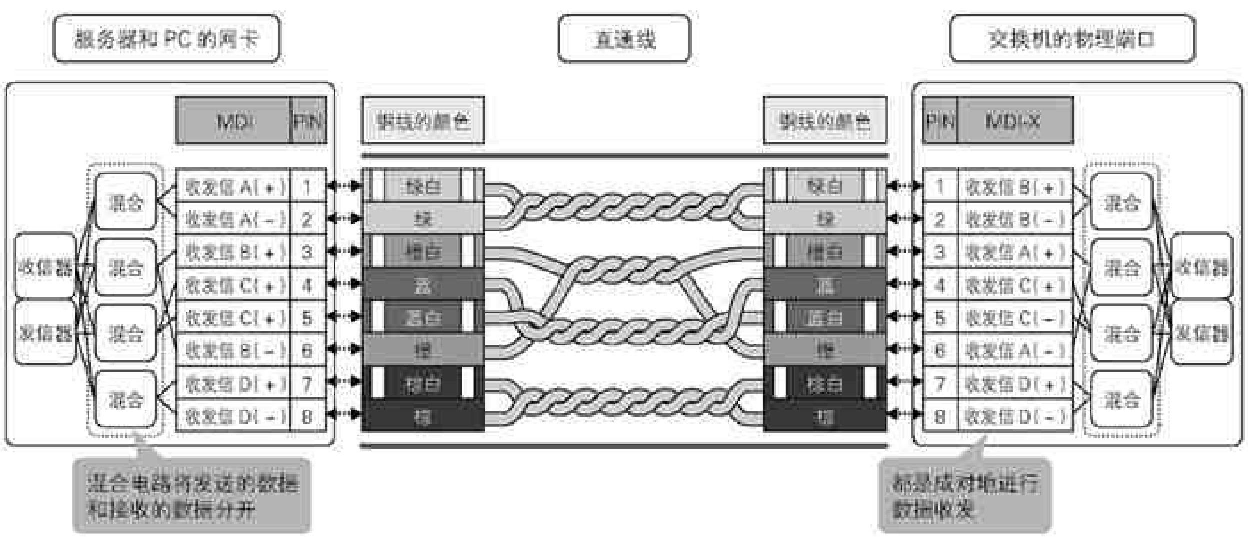


图 **1.1.11**　在 **1000BASE-T** 中八根铜线都会用到

相邻机器的速率和双工设置应保持一致

至此，我们对双绞线电缆繁琐而又深奥的结构做了说明。不过，物理层

中受制于线缆和网卡这些硬件的地方很多，实际设置网络设备时未必有那么多的设置细分项目。我们在使用双绞线电缆时只需注意两处设置即可，那就是端口的速率和双工。相邻设备的速率和双工设置请务必要保持一致。

正如其名，速率表示传输的速度。我们应配合服务器网卡和网络设备的物理端口将它设置成 100Mbit/s、1000Mbit/s 等值。

双工是指双向通信的方式。在网络的世界里不存在单方通信，只有双向通信。双工表示的就是双向通信成立的方式，它分半双工（half duplex）和全双工（full duplex）两种。半双工通信在同一时刻只能进行单向通信，需要通过调换方向的方式才能使双向通信成立。以前人们用 10BASE2、10BASE5 这样的旧规格，但现在不会刻意设置成那样了。在 10GBASE-T 中，连半双工这个概念都已经不复存在了。这是因为如果采用半双工通信，在收发批量文件等情况时会发生错误，吞吐率无法提升。与此相对，全双工通信则是同时进行收发处理来使双向通信成立的，分别设有不同的信道发送和接收信息。目前，全双工已经成为唯一的标准通信方式，端口的设置必须符合全双工通信的条件才行。

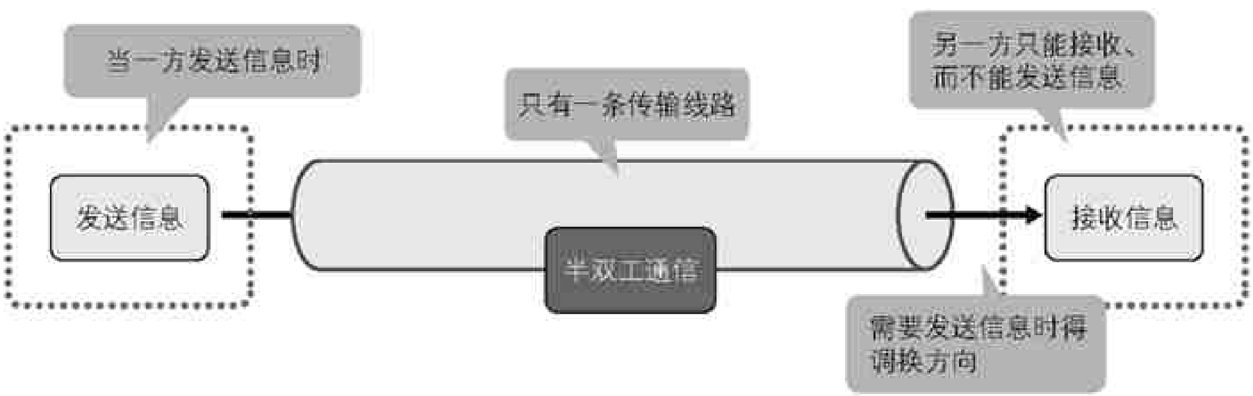


图 **1.1.12**　半双工通信只有一条传输线路

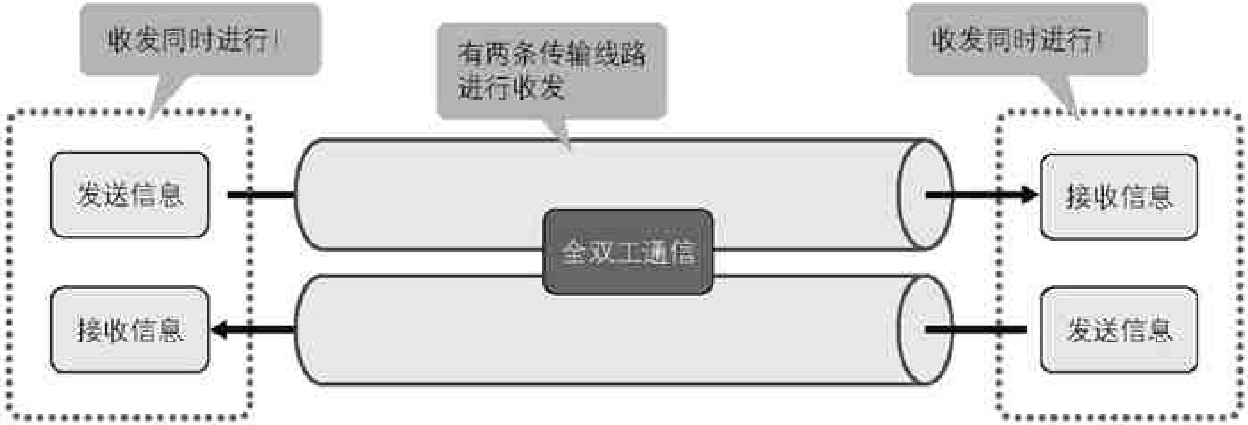


图 **1.1.13**　全双工通信有两条传输线路

速率和双工有手动设置和自动（自协商）设置两种设置方法，无论采用哪种方法，我们都要保持两端设备的设置一致。

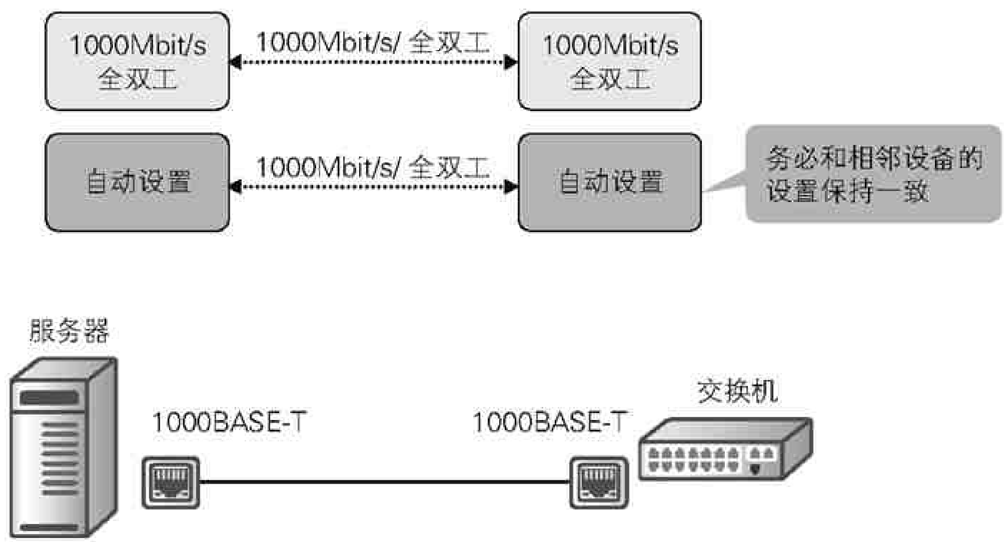


图 **1.1.14**　请务必确保两端设备的速率和双工设置一致

手动设置两端的端口时，速率如果设置得不一致，会连对接都无法完成；而双工如果不一致，则对接虽然能完成但基本上无法进行通信。

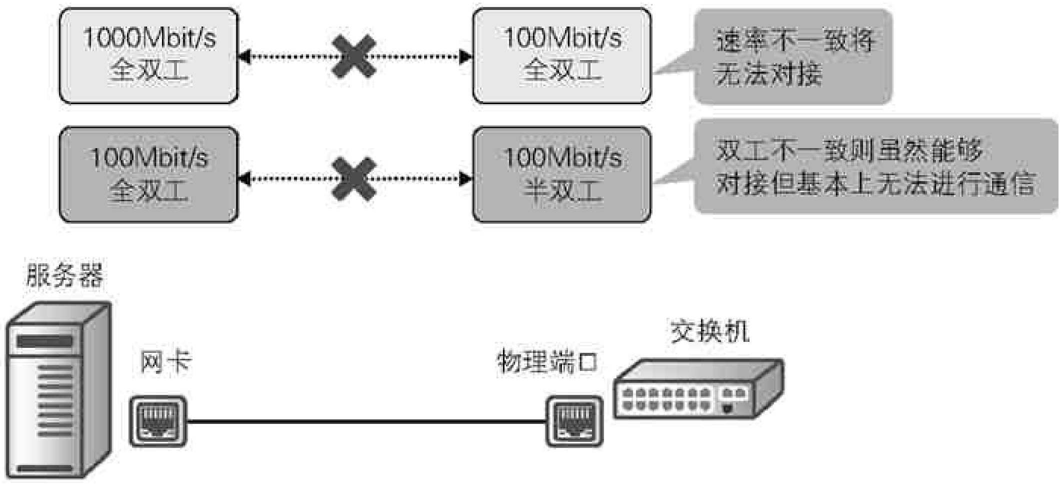


图 **1.1.15**　手动设置时如果速率或双工设得不一致将无法进行通信

采用自动设置（自协商）时也要注意。自动设置是通过交换 FLP（Fast Link Pulse，快速连接脉冲）信号来决定速率和双工的，通过 FLP 交换彼此支持的速率和双工，然后按照预先决定好的优先顺序来决定速率和双工。

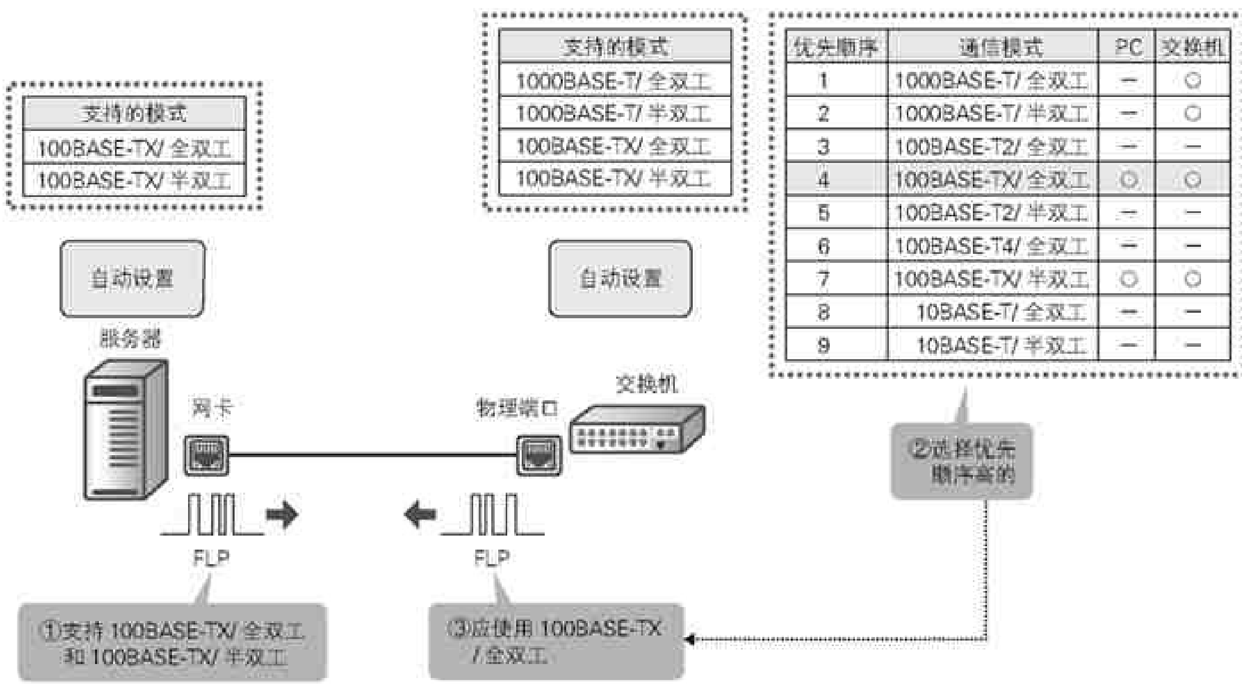


图 **1.1.16**　互相交换信号以决定速率和双工

当两端都采用自动设置时，结果必然是选择全双工，这没有什么问题。但是，如果只有一端采用了自动设置，那么系统将会选择默认的双工设

置——半双工。这是因为当发送的是 FLP 信号而返回的却是非 FLP 信号时，系统就会选择半双工模式。这时候吞吐率是提升无望的，所以如果一端采用了自动设置，另一端就一定也要采用自动设置，这个原则我们绝对要遵守。

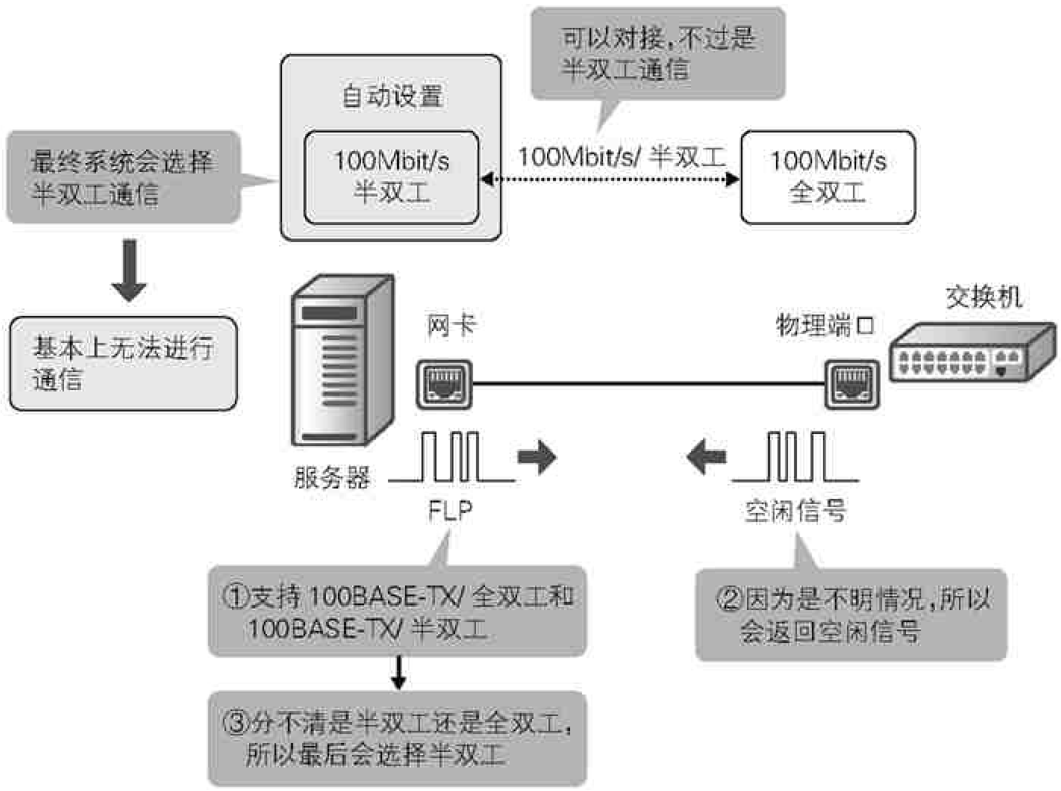


图 **1.1.17**　采用自动设置时要注意！

双绞线电缆的最大传输距离为 **100** 米

双绞线电缆是目前最常用的布线材料，但它有一个致命的弱点，那就是传输距离非常有限。从规格上来说，双绞线电缆最多只能延伸到 100 米处，如果延伸到 100 米之外，电信号会衰减，数据会消失，所以超过 100 米时必须设置交换机等中转机器以延长传输距离。布线时考虑到这个最大传输距离是极其重要的。有人也许会觉得有 100 米足够了，但是电缆需要装入导管，布线时会遇到障碍物，由于建筑物的某些原因而不得不迂回布线的情况也很多，你会意外地发现 100 米其实很短。所以我们应仔细确认布线线路，将总长度控制在 100 米之内。当然，你也可以通过中转机器让它得以无限延伸。不过那样的话，今后需要进行运行管理的机器将会增加，故障点也会增加，总体来说并不实用。遇到这种情况，我们应使用光纤光缆，至于光纤光缆，将在下个小节为大家说明。

用中继集线器抓包

在物理层设备中，最具实用性的就是中继集线器了。中继集线器的功能非常简单，它将端口接收到的信号复制并发送给所有端口。由于是将数据发送给所有端口，所以与该中继集线器连接的所有端口都会接收到和自己并不相干的数据，从通信量的角度来说这样做的效率并不高。最近，中继集线器大多已被交换集线器所替代，几乎很少能见到了，作为支持数据通信的集线器这个角色，我们可以认为它已经走到了终点。

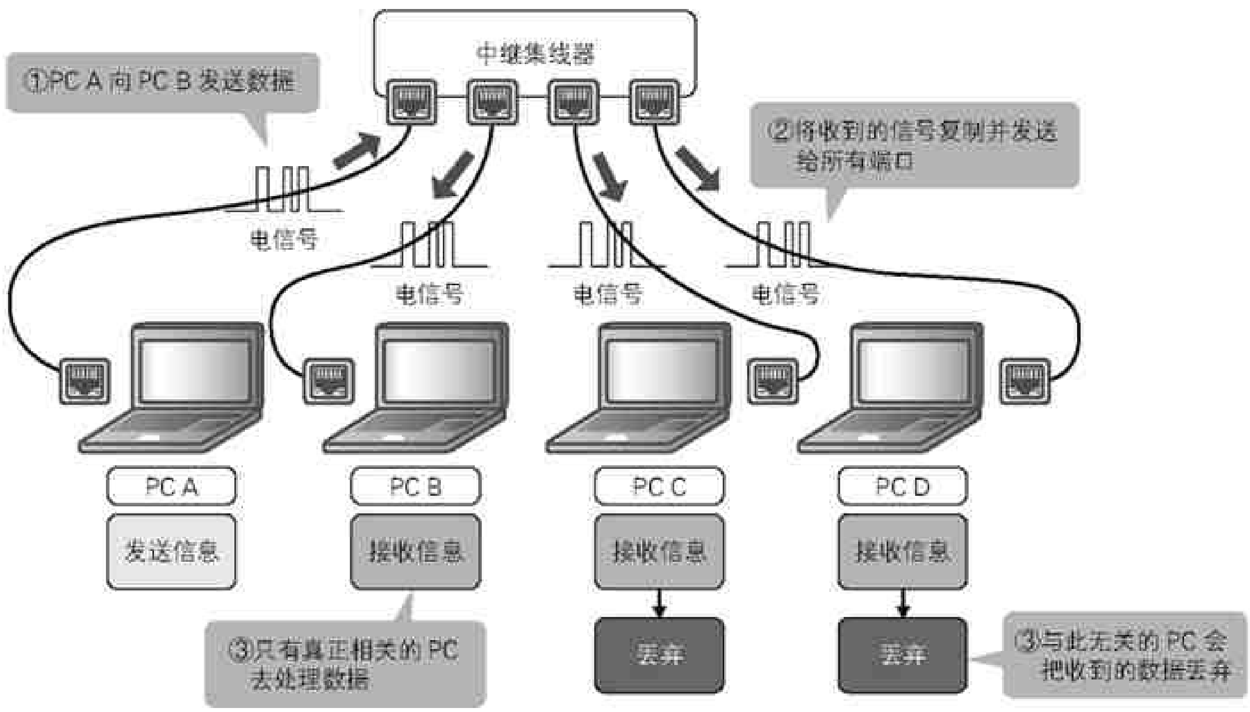


图 **1.1.18**　中继集线器将复制的信号发送给所有的端口

中继集线器的真本领在于故障排除。我们在需要抓包的 PC 或服务器之间插入中继集线器，然后在同列中配置好另一台 PC，就能接收到机器之间交换的信号。接收到的数码信号可用 Wireshark、EtherPeek 这样的专用软件进行截取和分析。使用中继集线器时能够顺利抓包而不改变现有机器的设置。因此，在细化到比特流程度的故障排除中，它总是能够发挥自己真正的本领。

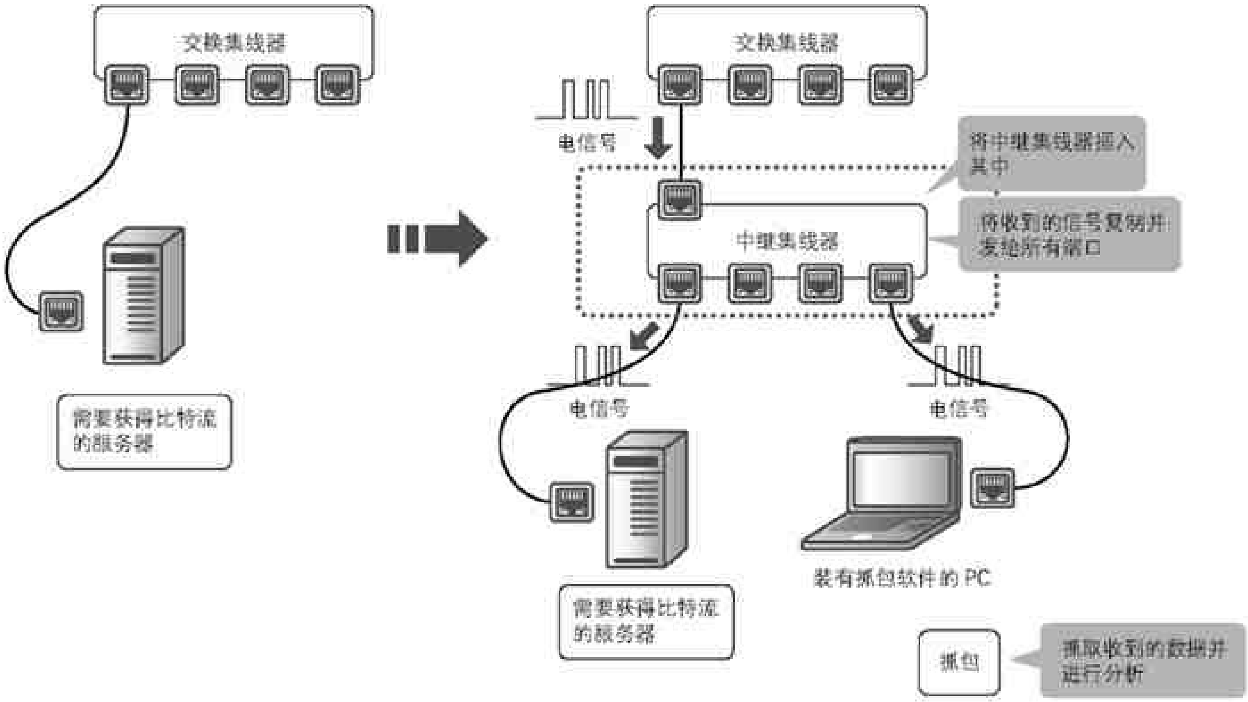


图 **1.1.19**　通过中继集线器排除故障

1.1.1.3　光纤光缆是用玻璃制成的

接下来，我们来看光纤光缆。光纤光缆的规格为○○ BASE-SX/SR 或○○ BASE-LX/LR。○○ BASE-SX/SR 中的 S 是 Short Wavelength（短波长）中的 S，○○ BASE-LX/LR 中的 L 是 Long Wavelength（长波长）中的 L，它们都代表了激光的种类。我们使用的激光种类将直接影响传输距离和所用的线缆。

光纤光缆是用极细的玻璃管做成的 [1]，用于传输光信号。光纤光缆由折射率较高的纤芯和折射率较低的包层构成，为同芯双层结构。通过将折射率不同的两种玻璃做成双层构造 [2]，可以将光封闭到纤芯内，进而形成低损耗的光通道，这个光通道就叫作模。

[1] 最近也出现了塑料光纤、高分子光纤等用非玻璃材质制成的光纤，不过人们一般使用的光纤电缆都是用高纯度的石英玻璃制成的。

[2] 如果算上被覆表层就是三层构造。

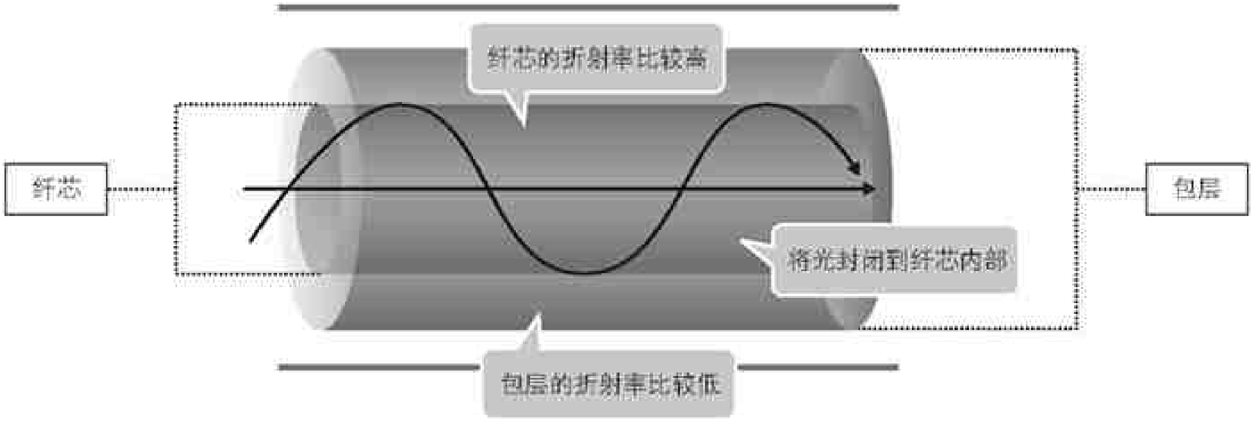


图 **1.1.20**　光纤光缆由纤芯和包层构成

实际传输数据的时候，一根芯用于发送信息，另一根芯则用于接收信息，两根芯组成一对，形成全双工通信[3]。由于必须构成收和发的关系，如果一方发送了信息，另一方就必须接收信息。如果双方都发送信息或都接收信息，就会连对接都无法完成。

[3] 1000BASE-BX是在一根芯中将波长分为用于发信的波长和用于收信的波长两种，分别进行信息收发的。

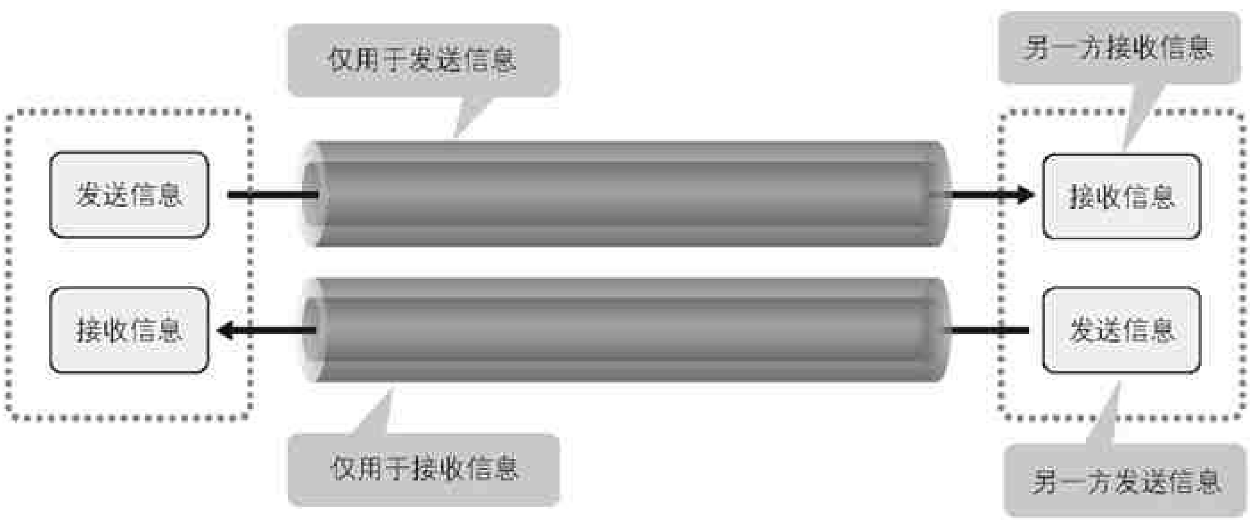


图 **1.1.21**　收发信息分别使用不同的光纤光缆

使用光纤光缆时，即使传输距离很长信号也不会衰减，能够保持极宽的频带，和双绞线电缆相比能延伸到更远的距离。但是光纤光缆也有缺点，那就是结构过于精密，使用起来不太方便。

根据不同需要区分使用多模光纤和单模光纤

光纤光缆有多模光纤（MMF）和单模光纤（SMF）两种类型，二者的差异在于纤芯直径（即芯径，光信号从这里通过）不同。

- 多模光纤

  多模光纤的芯径为 50 μm 或 62.5 μm，用于 1000BASE-SX 和 10GBASE-SR[4] 。由于芯径较大，光的通道（模）分散成了好几个（多）。由于有多个通道，和单模光纤相比，它的传输损耗更大，传输距离更短（最大 550 米）。然而它的价格比单模光纤便宜，也更好用，因此常常用于 LAN 这种比较近距离的传输。多模光纤按纤芯折射率可分为 SI 型（阶跃）和 GI 型（渐变）两种，目前人们使用的多模光纤一般为 GI 型，我们可以认为多模光纤指的就是 GI 型光纤。GI 型光纤能使纤芯的折射率逐渐发生变化，使所有光通道在同一时间到达目的地，这样就能减少传输损耗。

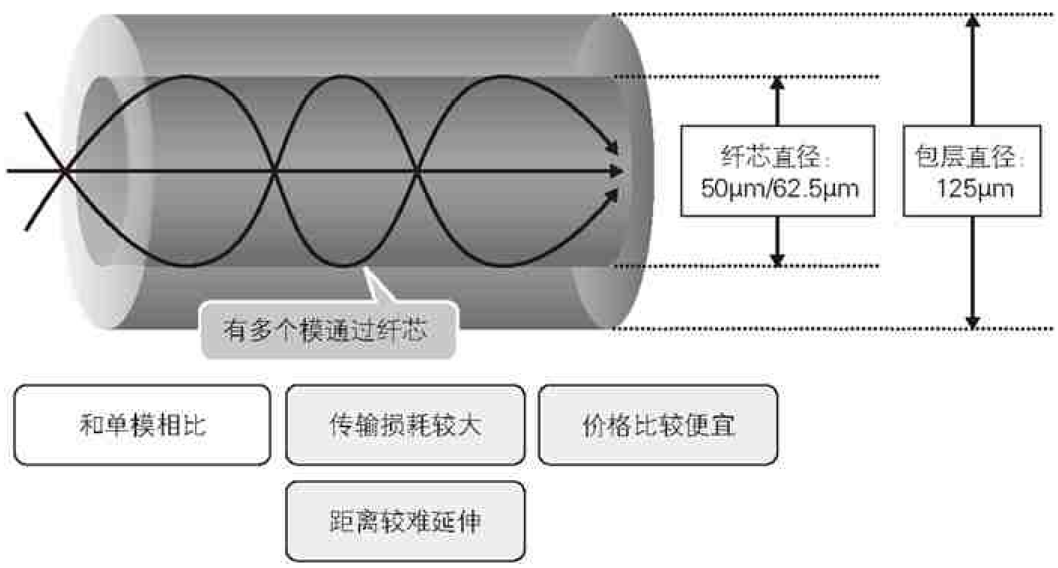


图 **1.1.22**　多模光纤中有多条光通道

- 单模光纤

  单模光纤的芯径为 8 ～ 10 μm，用于 1000BASE-LX 和 10GBASE-LR。通过缩小芯径，以及湿度控制纤芯和包层之间的折射率差，达到了只有一个（单）光通路（模）的效果。由于设计上保证了只有一条光通路存在，所以单模光纤能够实现长距离和大容量数据的

传输。单模光纤在家庭或公司等场所不易见到，但如果你经常出入数据中心或 ISP 主干网设施就会经常看到了。不知为何，笔者个人总觉得单模光纤以黄色的线缆居多。单模光纤在传输损耗小又能进行长距离通信这两点上无可挑剔，但缺点是价格偏高。

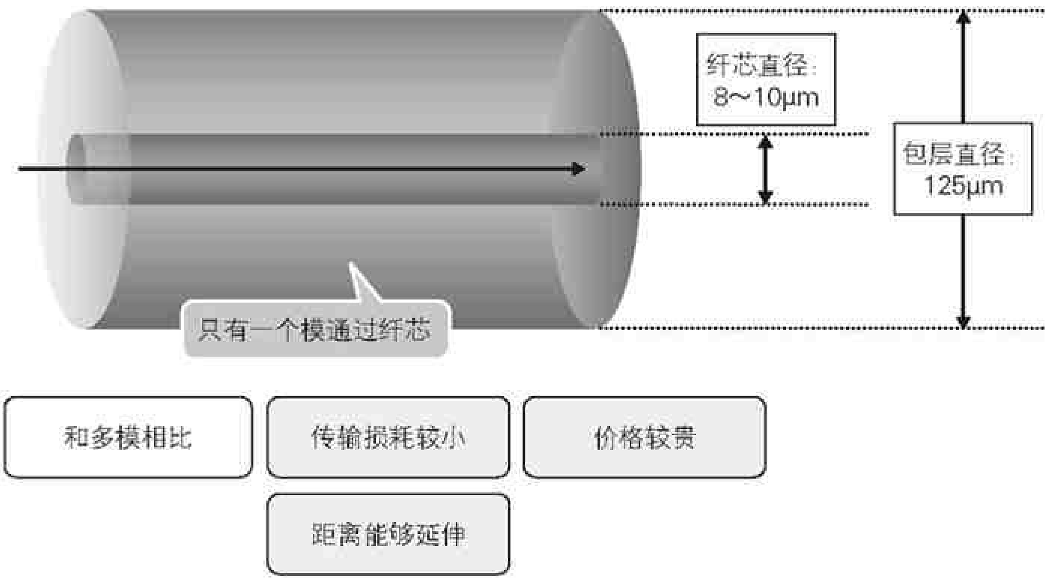


图 **1.1.23**　单模光纤中只有一条光通道

[4] 也可用于 1000BASE-LX，此时其最大传输距离即为多模光纤的最大传输距离（550 米）。

下表是将两种光纤光缆进行比较的结果。

表 **1.1.4**　单模光纤和多模光纤的比较

比较事项	多模光纤	单模光纤
纤芯直径	50 μm 62.5 μm	8～10 μm
包层直径	125 μm	125 μm
光通道（模）	多个	一个

模分散	有	无
传输损耗	小	更小
传输距离	最大 550m	最大 70km
使用	不方便	更不方便
成本	高	更高

光纤光缆的规格

光纤光缆有多种规格，常听大家说对这个不太了解。光纤光缆的具体规格直接关系到我们对它们的选择，所以在这里要好好整理一下。

我们在整理规格的时候，最容易理解的是○ BASE- □△中□的部分。这部分的文字表示了激光的种类，笔者就是利用这里的英文字母整理规格的。以 1000BASE-SX 为例，1000BASE-SX 中的 S 代表了 Short Wavelength（短波长）[5]，有 S 的规格说明该光缆使用的是 850nm 波长的短波长激光，仅能用于多模光纤。笔者是这样简单记忆的：S 代表 Short，指距离短的多模；而 L 代表 Long，指距离长的单模，不过大能兼小，所以它也能用于多模光纤。

> [5] X 则是 1000BASE-X 中的 X。1000BASE-X 是使用光纤的千兆位以太网的总称。顺便提一句，10GBASE-SR 中的 R 指 10GBASE-R 中的 R，而 10GBASE-R 是 10 千兆位以太网的总称。

下面是笔者整理出来的光纤光缆规格一览表。

表 **1.1.5**　可用于 **1000BASE-X** 的光缆

	传输媒介			
	MMF（多模光纤）		SMF（单模光纤）	
纤芯直径	62.5 μm	50 μm	8～10 μm	
距离※	275m（200MHz・km 时）	550m（500MHz・km 时）	10km	70km
波长	850nm		1310nm	1550nm
1Gbit/s 的规格（1000BASE-X）	1000BASE-SX IEEE802.3z			1000BASE-ZX 思科公司特有
	1000BASE-LX IEEE802.3z			

※ 括号内的数字表示模的带宽，数字越大则传输损耗越小，适合长距离传输。

表 **1.1.6**　可用于 **10GBASE-R-X** 的光缆

	传输媒介			
	MMF（多模光纤）		SMF（单模光纤）	
纤芯直径	62.5 μm	50 μm	8～10 μm	
距离※	33m（200MHz・km 时）	300m（2000MHz・km 时）	10km	70km
波长	850nm		1310nm	1550nm
10Gbit/s 的规格（10BASE-R）	10GBASE-SR IEEE802.3ae		10GBASE-LR IEEE802.3ae	10GBASE-ER IEEE802.3ae

※ 括号内的数字表示模的带宽，数字越大则传输损耗越小，适合长距离传输。

常用的连接器有 **SC** 型和 **LC** 型两种

光纤连接器有多种形状，一般用于 LAN 等方面的连接器有 SC 型和 LC 型两种，我们应根据连接的设备和模块选择使用哪一种。

- **SC** 型连接器

  SC 型连接器只需推入插头就能锁住，向外轻拉就能断开，是一种推拉式构造的连接器。其特点是使用方便、成本低廉，缺点是插头有点大。在 10GBASE-R 中使用的 X2 模块、将电信号和光信号相

互转换的媒体转换器、ONU（Optical Network Unit，光网络单元）设备……连接这些对象时要用到 SC 型连接器。



图 **1.1.24　SC** 型连接器的形状（照片提供源：**SANWA SUPPLY** 股份有限公司）

- **LC** 型连接器

LC 型连接器的形状和 SC 型很像。它和双绞线电缆的连接器（RJ-45）一样只需推入插头就能锁住，要断开时按住小突起（闩锁）往外拉即可。插头比 SC 型连接器的小，能够安装很多端口。连接 SFP 模块、SFP+ 模块时使用 LC 型连接器。



图 **1.1.25　LC** 型连接器的形状（照片提供源：**SANWA SUPPLY** 股份有限公司）

SC 型连接器和 LC 型连接器只是连接器的形状不同而已，和使用规格并没有直接的关系。首先我们应根据需要的规格选择连接的设备和模块，然后根据该设备和模块选择连接器和光缆。举个例子，有两台装交换机装有 1000BASE-SX 的 SFP 模块，当我们需要将这两台交换机连接起来的时候，SFP 模块应接 LC 型连接器，应选择 LC-LC 多模光纤。

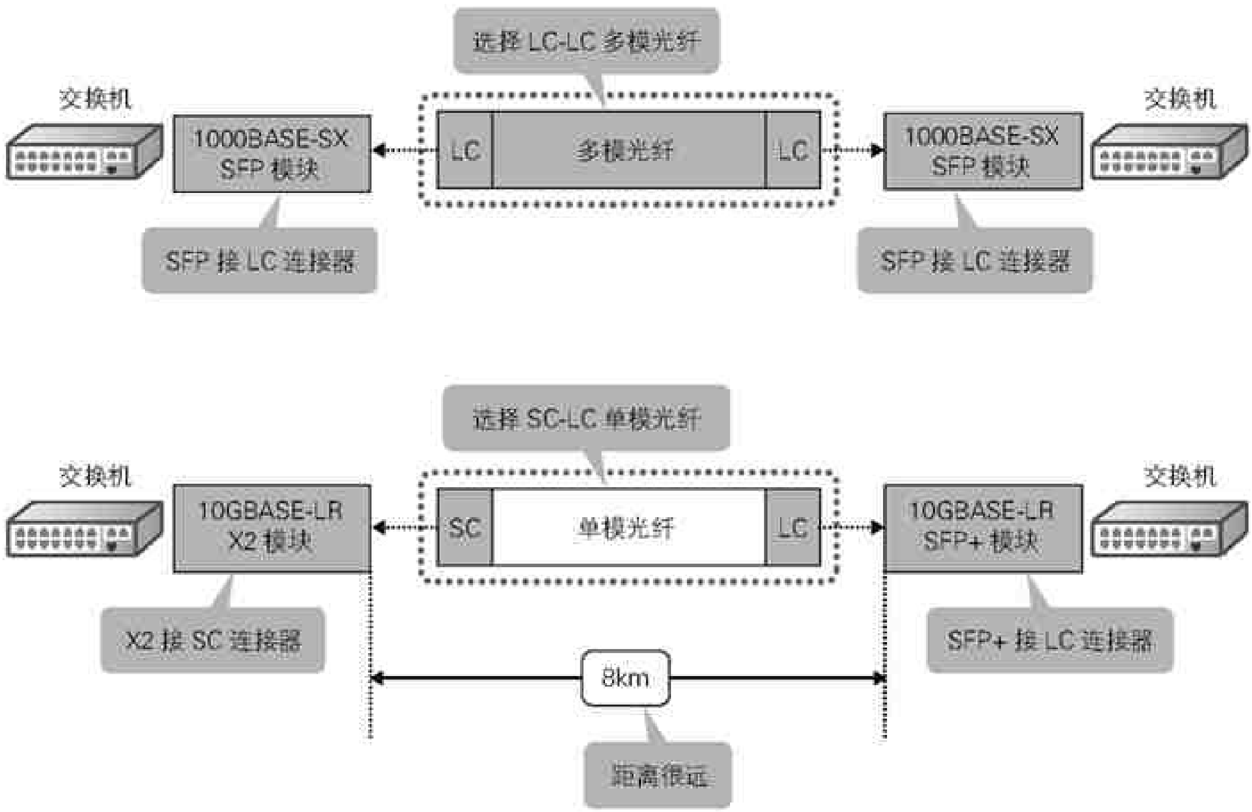


图 **1.1.26**　根据设备和模块选择连接器和光缆

1.2　物理设计

前面我们已经对物理层的各种技术做了很多说明。下面从实用性的角度讲讲在服务器端应该如何使用这些物理层的技术，以及我们在设计和架构服务器端时应该注意哪些事项。

1.2.1　服务器端有两种结构类型

想要设计出服务器端物理层面的结构，就要搞清楚要针对什么设备进行怎样的配置和连接。服务开始运作之后再去大规模地修改结构是非常困难的——当然并非无法实现，但是届时我们将不得不中断服务。因此，为了避免出现那样不得已的局面，我们必须设计出易于管理、易于扩展、着眼于未来的物理结构。

一般用于服务器端的物理结构有两种，分别是串联式和单路并联式，中小规模的系统环境多采用前者，大规模系统环境则多采用后者。下表是将二者进行粗略比较之后得出的结果。

表 **1.2.1**　串联式结构和单路并联式结构的比较[6]

[6] 表格中的○代表“高”，△代表“尚可”。——译者注

比较事项	串联式结构	单路并联式结构
结构的简易度	○	△
故障排除的简易度	○	△
结构的灵活性	△	○
可扩展性	△	○
冗余性与可用性	○	○

采用的系统规模	中小规模	大规模

关于各种机器的技术、功能，以及为什么是这样的物理结构等详细内容将在下一章按结构类型分别进行讲解。在这一节，大家只需对如何连接有个大致的印象就可以了。物理结构的设计是诸多技术和诸多功能的集大成者，所有的设备配置都自有其道理和意义，相信各位读者在了解了各种技术和功能之后再回过头来看这一节就会觉得更加有趣。接下来就开始详细介绍各种结构的概要。

1.2.1.1 采用串联式结构管理起来更方便

首先我们来看串联式结构。该结构是在通信线路上进行配置的，串联式（Inline）也因此而得名。目前在服务器端用得最多的就是串联式结构，由于它结构简单，故障也容易排除，深受运行管理者的喜爱。串联式结构有很多种类型，无法一一列举，这里只介绍几个有代表性的类型。

串联式结构之类型 **1**

我们来看第一种类型。这是串联式结构中最简单、最容易理解的一种结构，网络设备从上到下的配置都是四角形结构。这种方形结构是串联式结构中最基本的结构。

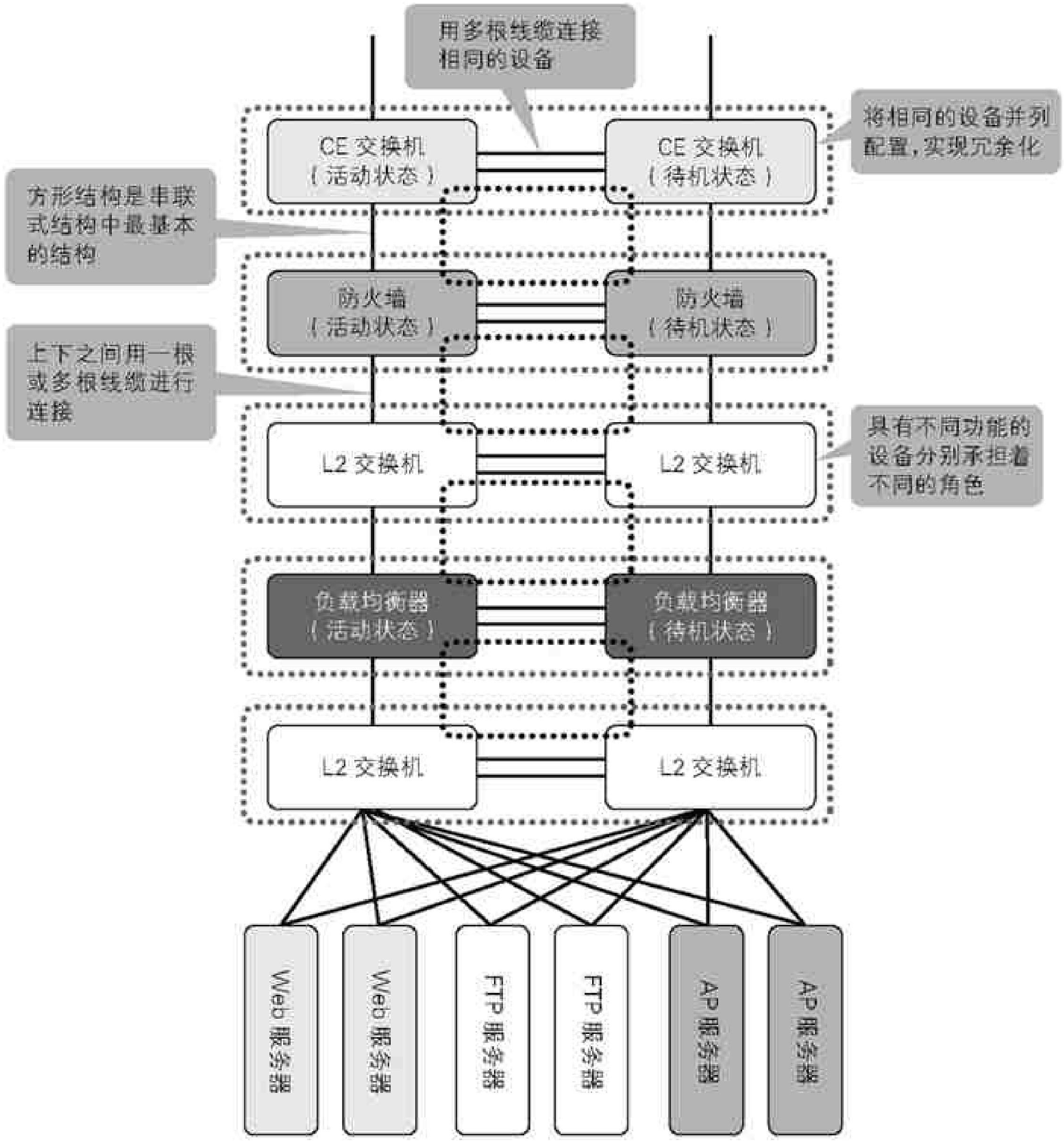


图 **1.2.1**　串联式结构之类型 **1**

我们应将相同的设备并列配置以实现冗余化，用多根线缆连接机器。这些线缆不仅交换管理比特流（管理比特流用于管理冗余结构），同时又提供发生故障时的迂回线路。上下之间用一根线缆连接。实际上也可根据通信量适当地增加线缆，但这里为了方便大家理解，只用一根。

这种结构非常简单。在这种结构中每台机器各司其职，无论哪台机器发生故障，系统都会将该故障对其他机器的影响控制在最小范围内，同时切换连接到备用机上。

串联式结构之类型 **2**

接下来看看第二种类型。这种类型是在第一种结构类型中加入了刀片服务器、虚拟化、StackWise 技术和 VSS（Virtual Switching System，虚拟交换系统）、数据安全区段划分这四个元素。乍一看比较复杂，但仔细观察后会发现它是一种非常典型的方形结构。这里，使用 StackWise 技术的交换机必须用专用的 Stack 线缆连接。

这种结构的刀片服务器和虚拟化能够提高服务器的集中效率和可扩展性，StackWise 技术和 VSS 能够提高交换机的运行管理效率，这些都对简化结构起着重要的作用。此外，对 DMZ 和 LAN 的数据安全区段划分又提高了安全度。这种类型必须采用刀片服务器特有的结构，架构可使用虚拟化功能的专用网络，较之类型 1，需要考虑的地方会更多一些。本书后面有对各个要素的详细说明，刀片服务器在 4.1.1.2 节，虚拟化在 2.3.1.1 节，StackWise 技术和 VSS 在 4.1.1.3 节，数据安全区段划分在 3.1.1.1 节。

这种结构也和类型 1 一样，每台机器各司其职，无论哪台机器发生故障，系统都会将故障对其他机器的影响控制在最小范围内，同时切换连接到备用机上。

1.2.1.2　采用单路并联式结构更容易扩展

下面，我们来看看单路并联式结构。这种结构恰似将设备配置到核心交换机的手臂位置，所以叫作 One-Arm 结构（单臂结构或单路并联式结构）。位于服务器端中心部位的核心交换机扮演着多重角色，因此整体结构比串联式结构要复杂。然而这种结构能够适应各种需求，兼具灵活性和可扩展性，常用于数据中心、多租户环境等规模较大的服务器端。下面介绍几个具有代表性的单路并联式结构类型和它们各自的特点。

单路并联式结构之类型 **1**

我们来看第一种类型。这是单路并联式结构中最简单、最容易理解的一种结构，实际上，这种结构在逻辑上和串联式结构的类型 1 是一样的。将相同的设备并列配置以实现冗余化这一点没有任何变化，不同的只是按纵列配置而已。在这种结构中，我们将防火墙和负载均衡器配置到核心交换机的旁边，就好像核心交换机的手臂一样。在串联式结构中，每台机器各司其职，配置非常容易理解；而在单路并联式结构中，很多职

能都集中于核心交换机一身，位于系统中心部位的核心交换机肩负着多重任务，几乎所有的通信都需要经过它的处理。

设备配置虽然完全不同，但单路并联式结构所用的冗余功能却和串联式结构的一模一样。无论哪台机器发生故障，系统都会立刻切换线路并确保新的线路畅通无阻。核心交换机由于承担着多重角色，一旦它自身发生故障，就会对所有的机器产生影响。

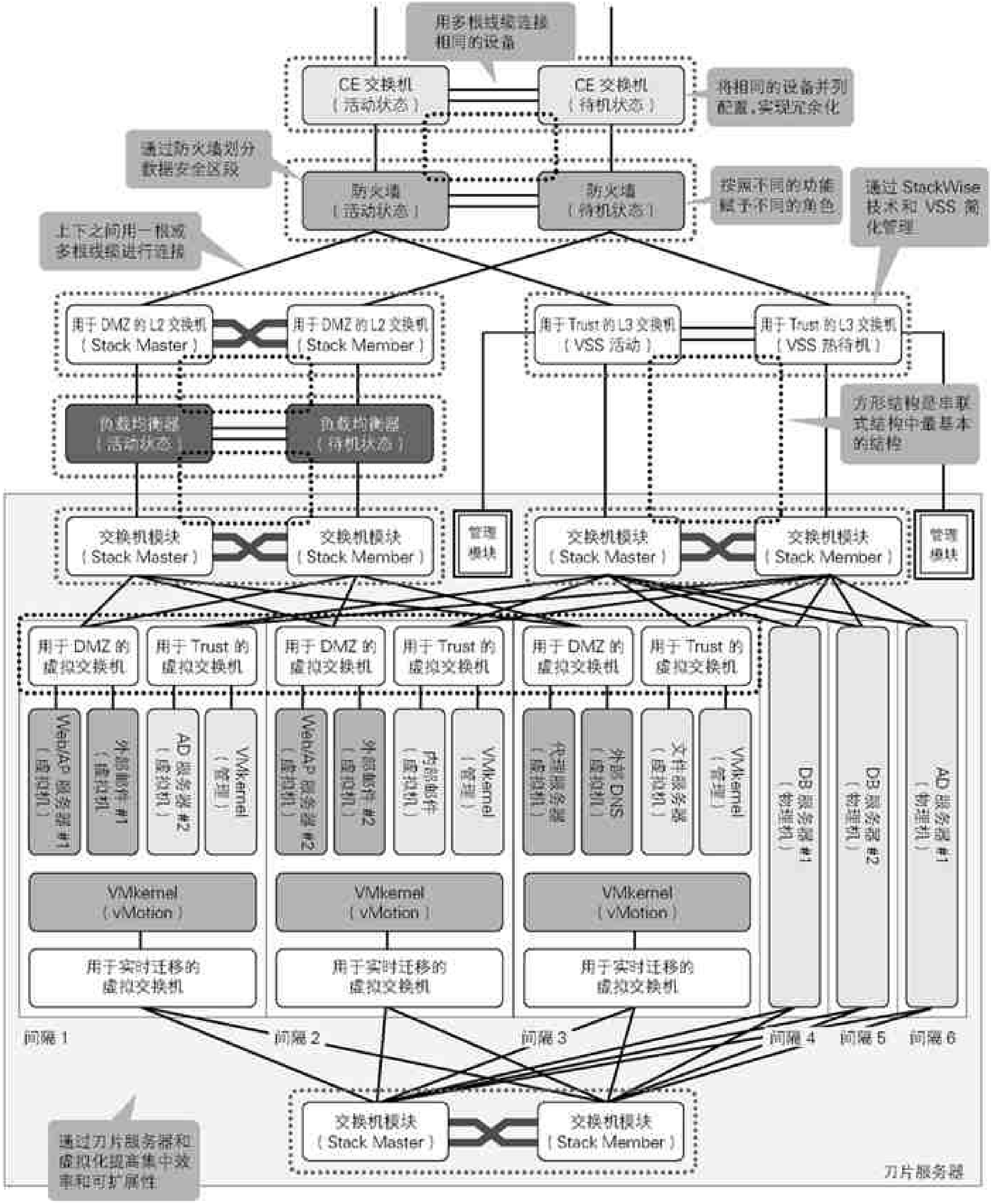


图 **1.2.2**　串联式结构之类型 **2**

※ 由于版面关系，部分服务器在图中省略未画。

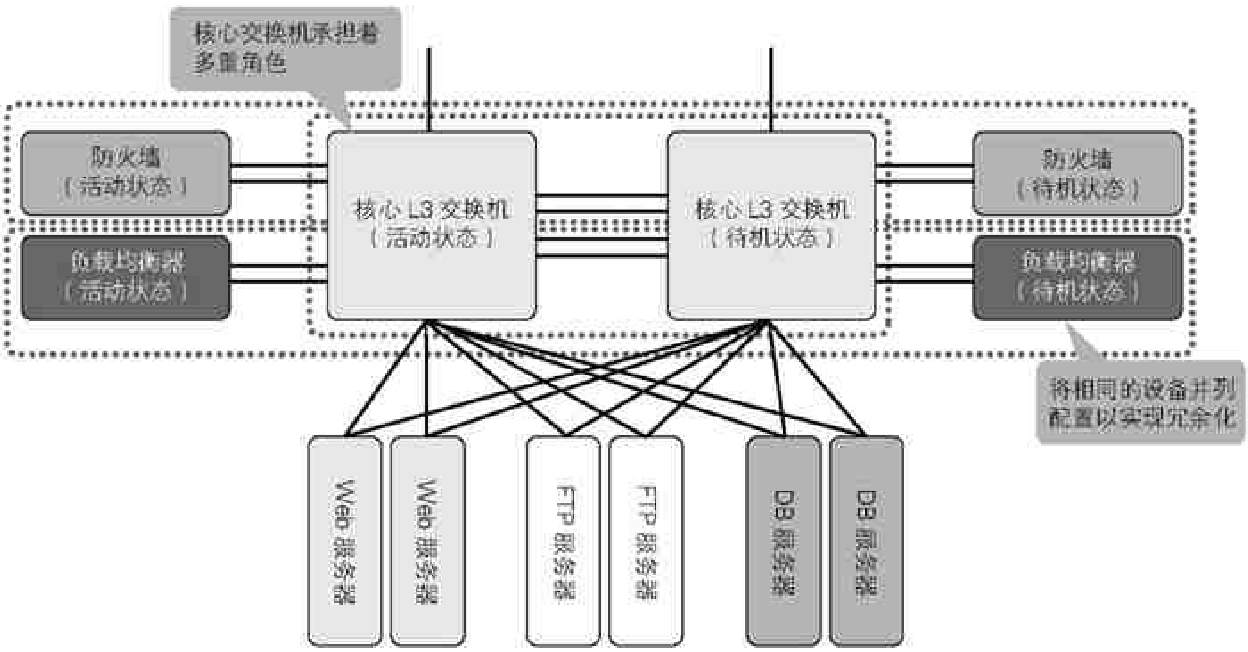


图 **1.2.3**　单路并联式结构之类型 **1**

单路并联式结构之类型 **2**

我们来看第二种类型。和串联式结构一样，这种类型是在第一种单路并联式结构类型中加入了刀片服务器、虚拟化、StackWise 技术和 VSS、数据安全区段划分这四个元素。

这种类型也是乍看起来比较复杂，但基本结构和结构类型 1 是一样的。核心交换机的连接形态稍有不同，这是因为 VSS 在逻辑上将两台核心交换机合并成了一台的缘故。从防火墙和负载均衡器的角度来看，就像是只连接到了一台核心交换机上那样，而它们就是配置在这台核心交换机旁边的手臂。

这种结构也和结构类型 1 一样，机器配置虽然完全不同，但所用的冗余功能却和串联式结构的一模一样。无论哪台机器发生故障，系统都会立刻切换线路并确保新的线路畅通无阻。这种结构的核心交换机和结构类型 1 的核心交换机一样承担着多重角色，不过，由于它是 VSS 在逻辑上合并成的一台交换机，所以即使它发生故障，也不会像类型 1 那样对周边机器产生巨大影响。

在之后的章节中介绍技术和规格时还会提到上述四种结构。在当前阶段这些结构看起来非常难懂，尤其是结构类型 2，里面糅合了太多内容，

现在大家也许还完全摸不着头脑。不过，一旦理解了后面章节介绍的技术和规格之后再回过头来看这些结构类型时，各位一定会觉得豁然开朗的。

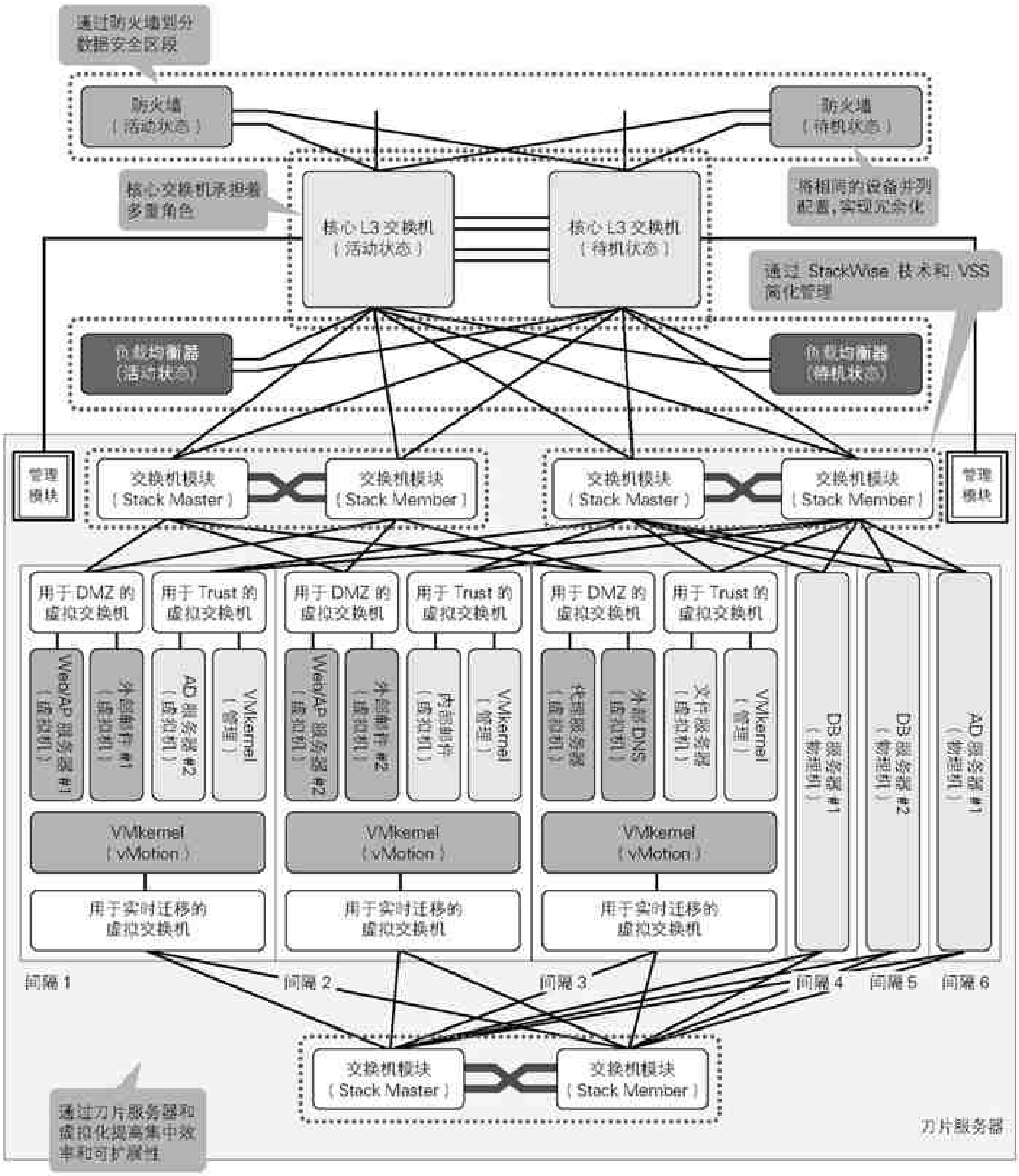


图 **1.2.4**　单路并联式结构之类型 **2**

※ 由于版面关系，部分服务器在图中省略未画。

1.2.2 选用设备时应参考考查项的最大值

设备的物理结构制定下来后，我们就要考虑应该选用有着怎样规格（性能）的设备。根据什么要素去选用设备，这就是硬件结构设计的内容。有时候我们可能还会需要另外准备一些考查项，它们并非物理设计的范畴，而是属于硬件结构设计和性能设计中的内容。本书立足于物理设备的性能设计来讲解物理设计。

选择设备时应考虑需要的功能、成本、吞吐率、连接器的数量、设备被认可和采纳的实际情况等多种要素。这里，我们着重针对吞吐率和连接器的数量进行说明。这两个要素可以成为我们选择设备时的绝对性指标。如果要更换旧机器，最好利用 SNMP（Simple Network Management Protocol，简单网络管理协议）预先从现有机器中获取这两个指标的相关数字，为今后选择设备做准备。关于 SNMP 将在本书的 5.1.2 节中详细介绍。如果要增设新机器，则应根据我们设想的用户人数、应用程序和信息内容大小等进行预测。

尽管这种情况屡见不鲜，但是像吞吐率和连接器数量这样的考查项，用它们一般的平均值去选择设备是没有多大意义的。我们应对长期或短期的访问类型进行分析，参考考查项的最大值去选择使用何种设备。

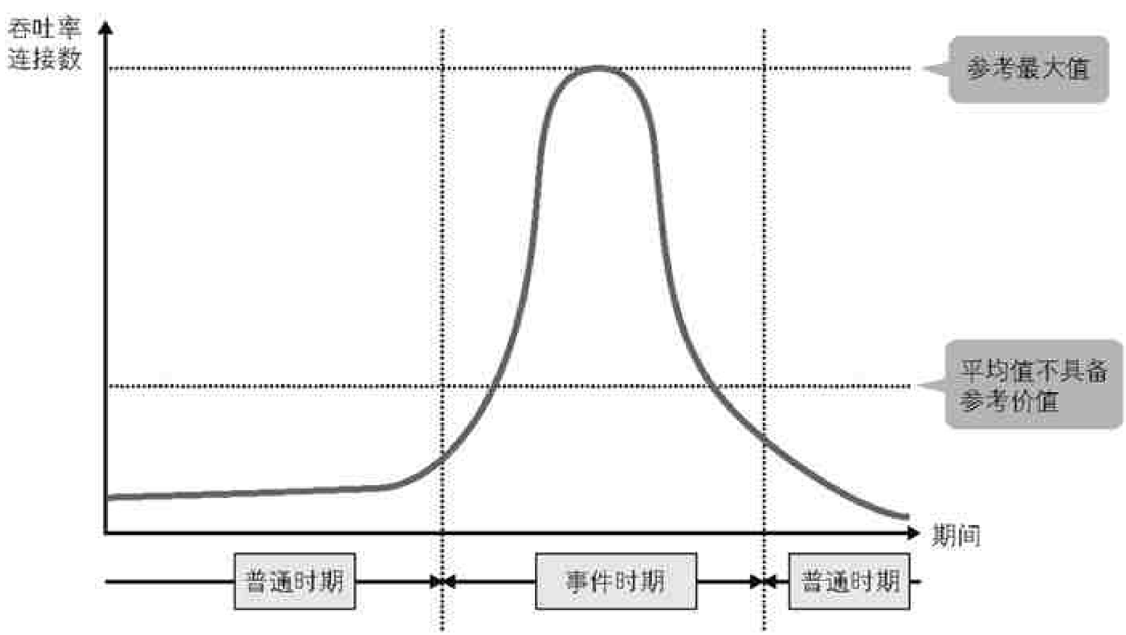


图 **1.2.5**　参考考查项的最大值

1.2.2.1　应用程序不同吞吐率也就不同

吞吐率指的是应用程序实际传输数据的有效速度。吞吐率中包含与应用程序相关的各种处理延迟，因此实际的传输速度肯定会比规格上的理论值要低。服务器端的吞吐率受诸多需求因素的影响，包括我们设想的最大连接用户数、所用应用程序的信息流通类型，等等。我们应在对这些因素了如指掌的基础上算出我们需要的吞吐率。网络设备的各生产商一般会公布其产品的最大吞吐率，即在不发生位丢失的情况下能够传输的最高速度。我们应参考那些数据，选择性能上仍有一定富余的机器。

有的设备会因为启用的功能导致最大吞吐率降低。遇到这种情况，我们应在充分考虑使用的功能之后再去决定机器的最大吞吐率。举个例子，有的设备在仅启动防火墙功能时处理速度可以达到 4Gbit/s，但同时开启 IPS（入侵防御系统）功能的话，处理速度就会降到 1.3Gbit/s。如果一定要用 IPS 功能则最大吞吐率应为 1.3Gbit/s，这个值才是我们应参考的对象。

1.2.2.2　新增连接数和并发连接数都要考虑

连接数是指一秒钟之内能够处理多少连接。连接数的值越大，表示能够处理的数据越多。选用防火墙和负载均衡器时我们需要注意这个数值。连接数分为新增连接数和并发连接数两种，新增连接数指每秒钟能够处理多少连接，并发连接数指同时能够维持多少连接。

这两个值看起来似乎应该成正比，但其实未必。例如，当 FTP（File Transfer Protocol，文件传输协议）这样的协议试图在长时间内维持少量的连接来进行访问时，并发连接数会增多，新增连接数却并不会增加多少。

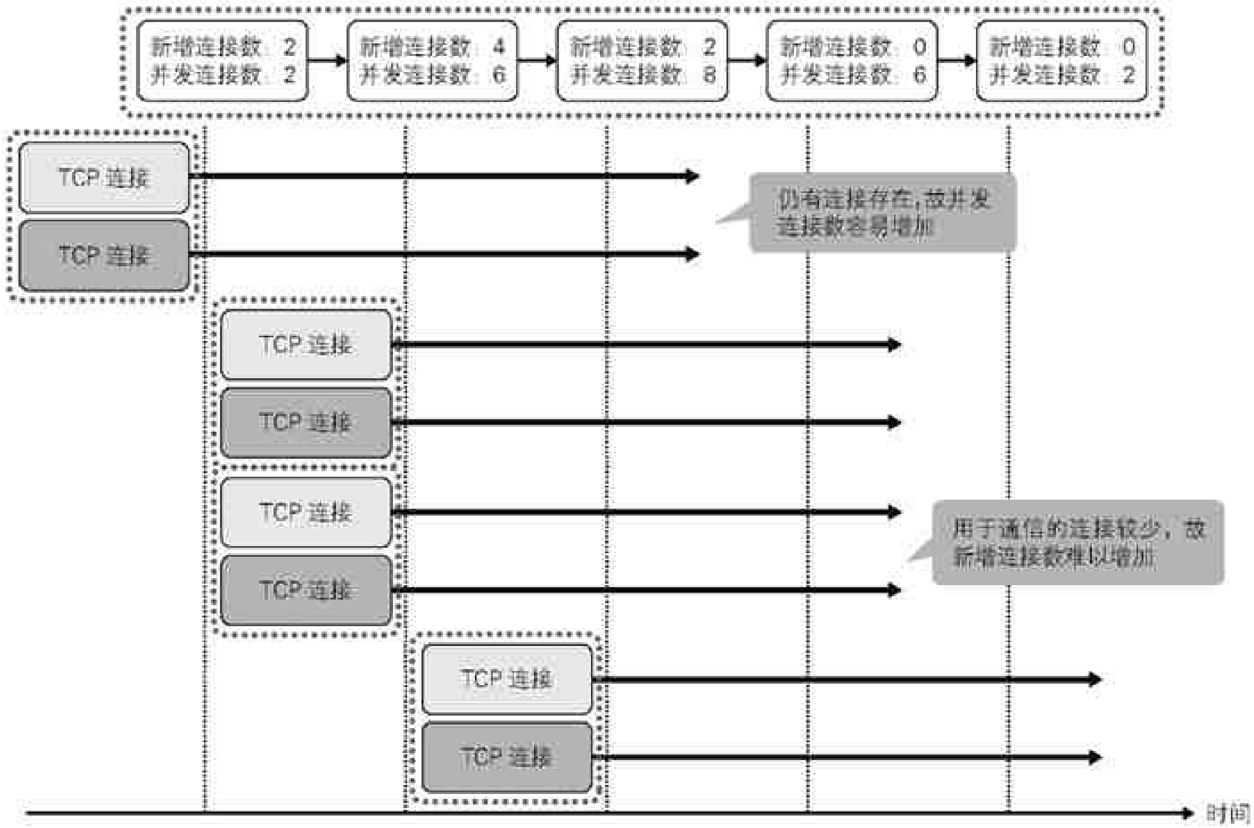


图 **1.2.6**　新增连接数和并发连接数未必成正比

我们设想的最大连接用户数和所用的应用程序会影响到我们实际需要的连接数。我们应明确了解这些因素之后再去计算所需的新增连接数和并发连接数。各生产商也会公布其产品的最大连接数[7]，我们应参考那些数据，选择性能上仍有一定富余的设备。无论是最大新增连接数还是最大并发连接数，实际的数字一旦超过它们，就会发生服务延迟。

[7] 有的生产商仅公布了并发连接数。

前面都是对硬件结构设计的说明，归根结底，设备是无法满足超出它们规格之外的要求的。无论我们使用多少系数去进行怎样的计算，得出的结果都是在我们能够设想到的范围之内。有些企业会按照极限值进行设定并根据该设定去选用设备，可是那样的话，一旦遇到出乎意料的巨大通信量，就会措手不及，根本无法应对。和服务器相比，网络设备的使用寿命一般更长一些，无论纵向扩展还是横向扩展都比较困难，所以我们在进行性能设计时一定要留有一定的余地才行。

根据什么信息去选用设备，这其中的逻辑真的十分重要。

1.2.3　选择稳定可靠的 OS 版本

网络设备和服务器一样都是在 OS 中运行的。然而不可思议的是，人们往往非常关心服务器的 OS 版本，对网络设备的 OS 版本却不是那么在意。其实，网络设备的 OS 版本也和服务器的 OS 版本同样重要。尤其是最近的网络设备，有些是在 Linux OS 中运行并在该 OS 的上层提供服务功能的，这就更需要我们在深思熟虑之后做出决定。如果采用了不稳定的 OS 版本，或是系统内出现了 OS 版本差异，那就可能会给今后的操作运行带来诸多的麻烦和障碍，所以我们一定要认真对待选择 OS 版本的问题。

不懂就问是捷径

网络设备使用的 OS 并不因为它是最新的版本或者执行了最新的修改程序（补丁、修补程序）就一定是稳定的。该 OS 被认可采纳的实际情况如何，生产商情况以及代理商的推荐起着重要的参考作用。有些生产商和代理商会公布他们的推荐版本和稳定版本。另一方面，也有些企业对企业内使用何种版本是有明文规定的，最好参考这些内容去挑选合适的 OS 版本。

OS 都是有漏洞的。系统运行后，生产商和代理商会通过简讯及时公布各种漏洞的存在，但我们不必去一一应对，否则会一直没完没了。我们只需要挑出那些和系统使用功能相关的漏洞以及会对系统产生较大影响的漏洞，对它们进行修补即可。此外还有一点，支持期限也可以成为我们挑选 OS 版本时的一个考查项。有些机器使用的每种 OS 版本都设有各自的支持期限，一旦超过该期限，官方将不再提供任何补丁。本来一直使用同一个版本这种事情就是不可能的，所以我们应该定期地检查当前正在使用的 OS 版本，不断进行版本升级。下面以负载均衡器的默认标准软件 BIG-IP 为例，在表 1.2.2 中列出了其各种版本的支持期限。

表 **1.2.2**　不同版本有不同的支持期限（截至 **2013** 年 **12** 月的数据）

软件版本	发布时间	软件开发结束时间	技术支持结束时间
11.4.1	2013 年 09 月 17 日	2016 年 09 月 17 日	2017 年 09 月 17 日

11.4.0	2013 年 06 月 11 日	2015 年 06 月 11 日	2016 年 06 月 11 日
11.3.0	2012 年 12 月 17 日	2014 年 12 月 17 日	2015 年 12 月 17 日
11.2.1	2012 年 09 月 25 日	2015 年 09 月 25 日	2016 年 09 月 25 日
11.2.0	2012 年 06 月 11 日	2014 年 06 月 11 日	2015 年 06 月 11 日
11.1.0	2011 年 11 月 28 日	2013 年 11 月 28 日	2014 年 11 月 28 日
11.0.0	2011 年 08 月 24 日	2013 年 08 月 24 日	2014 年 08 月 24 日
10.2.4	2012 年 04 月 11 日	2015 年 12 月 31 日	2016 年 12 月 31 日
10.2.3	2011 年 10 月 13 日	2015 年 12 月 31 日	2016 年 12 月 31 日
10.2.2	2011 年 06 月 17 日	2015 年 12 月 31 日	2016 年 12 月 31 日
10.2.1	2011 年 01 月 20 日	2015 年 12 月 31 日	2016 年 12 月 31 日
10.1.0	2009 年 12 月 17 日	2015 年 12 月 31 日	2016 年 12 月 31 日

* 参考网页：http://support.f5.com/kb/en-us/solutions/public/5000/900/sol5903.html?sr=28305237

1.2.4 根据实际配置和使用目的选择线缆

这里，我们要设计用于连接的网线，也就是要选择使用什么线缆进行连接。当前，对用于服务器端物理层的传输媒介我们有两个选择——双绞线电缆和光纤光缆。如今，无线 LAN 即使再高速，系统管理人员也不会跃跃欲试地想把它们用到服务器端去吧。在选择使用哪种传输媒介时，我们必须充分考虑成本、便利性、物理层面的配置以及数据用途等实际需求。这里，我们将着重针对物理层面的配置和数据用途进行说

明。

1.2.4.1 远距离传输选择光纤光缆

前面已经讲过，用于有线 LAN 的线缆都会受到传输距离的限制。距离越远信号强度就越低，数据的损耗率就越大，当然吞吐率也会下降。特别是双绞线电缆，由于仅能延伸到 100 米处，该现象尤为突出。如果需要跨越好几个楼层或是跨越大楼或房子，最后设备间的距离又接近 100 米的话，那么选择光纤光缆会比较安全。如果需要连接到更远的地方，则建议选择单模光纤光缆。我们应该根据传输距离选择传输媒介。

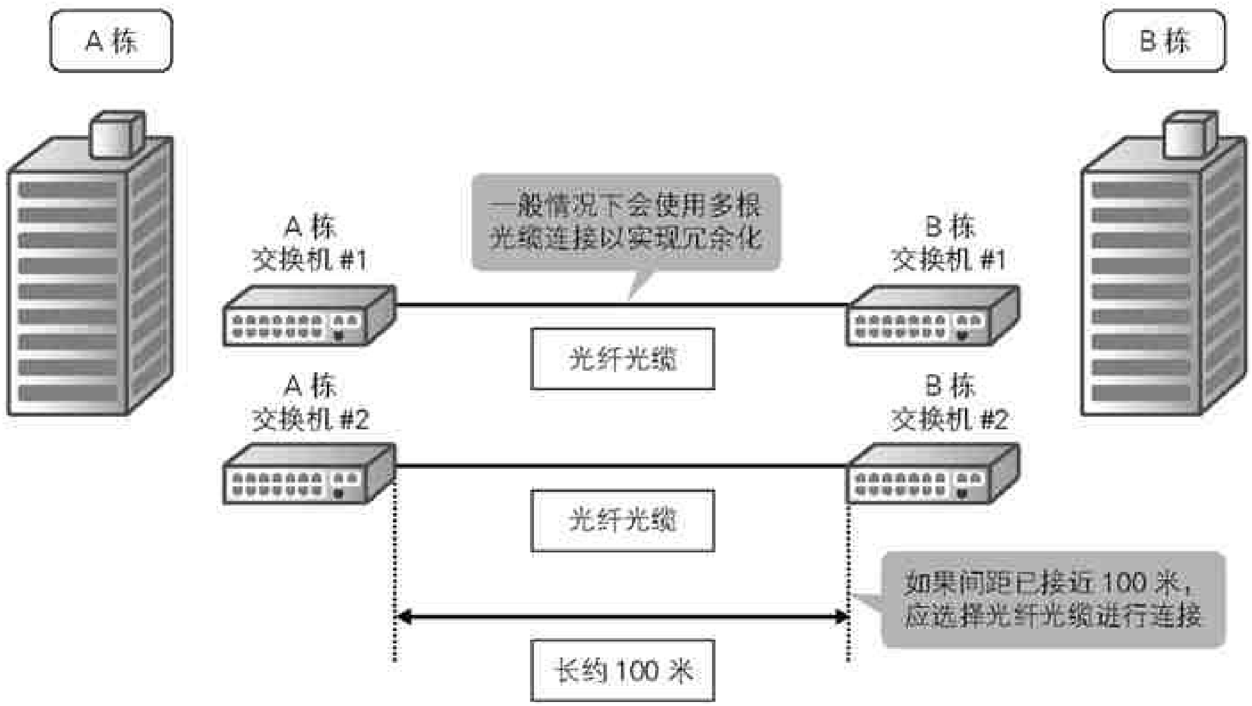


图 **1.2.7** 如果间距已接近 **100** 米，应选择光纤光缆进行连接

当交换机并不支持光纤光缆却需要延伸传输距离的时候，我们可以用媒体转换器来解决这个问题。媒体转换器是将双绞线电缆中流通的电信号转换为能在光纤光缆中流通的光信号的机器。首先，我们将不支持光纤光缆的设备通过双绞线电缆连到媒体转换器上；接下来，用 SC-SC 光纤光缆将其续接到另一台媒体转换器上；最后，我们再次使用双绞线电缆将媒体转换器连接至另一台设备。这样，传输距离就得以延伸了。由于媒体转换器只能使用 SC 型光纤连接器，所以这种情况时我们需要使用 SC-SC 光纤光缆。

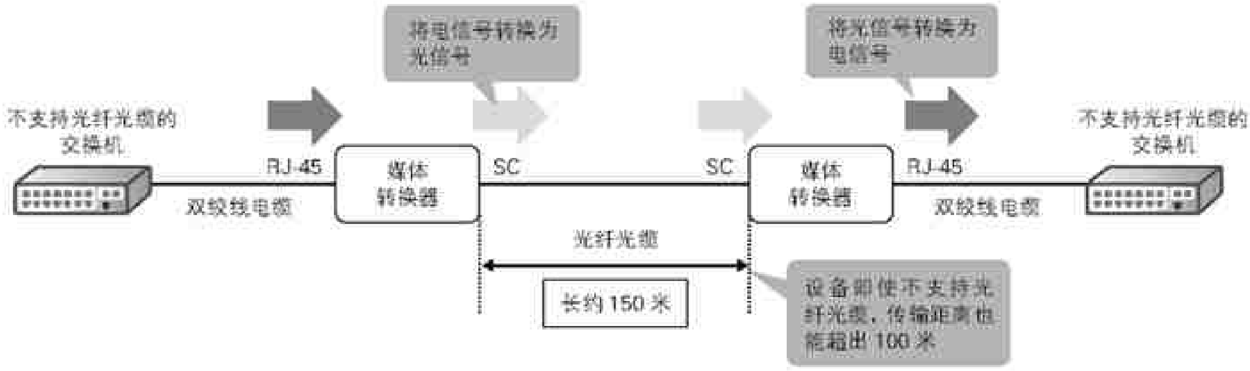


图 **1.2.8**　利用媒体转换器来延伸距离

1.2.4.2　追求宽频带和高可靠性时选择光纤

双绞线电缆在高频下的信号衰减十分明显，所以用于宽频（高速）传输的话性能有限。最近，通信统一已成为一种趋势，各种通信类型都开始在网络中出现。就好像三条车道比一条车道好用、道路越宽敞就越不容易发生交通拥堵那样，如今，宽带传输已成为了系统设计中不可或缺的内容。像 iSCSI、FCoE（Fiber Channel over Ethernet，以太网光纤通道）等存储器通信那样既需要始终维持宽带、又追求高可靠性的通信，使用光纤光缆比较合适。另外，通往位于上层的核心交换机、汇聚交换机 [8] 等交换机的上行部分也容易汇集大量的通信数据，因此也同样需要宽带和高可靠性。在这些地方我们应使用光纤光缆以确保宽带和高可靠性。

[8] 汇聚交换机是指将接入交换机和服务器交换机集中到一处的交换机，也叫分布交换机。

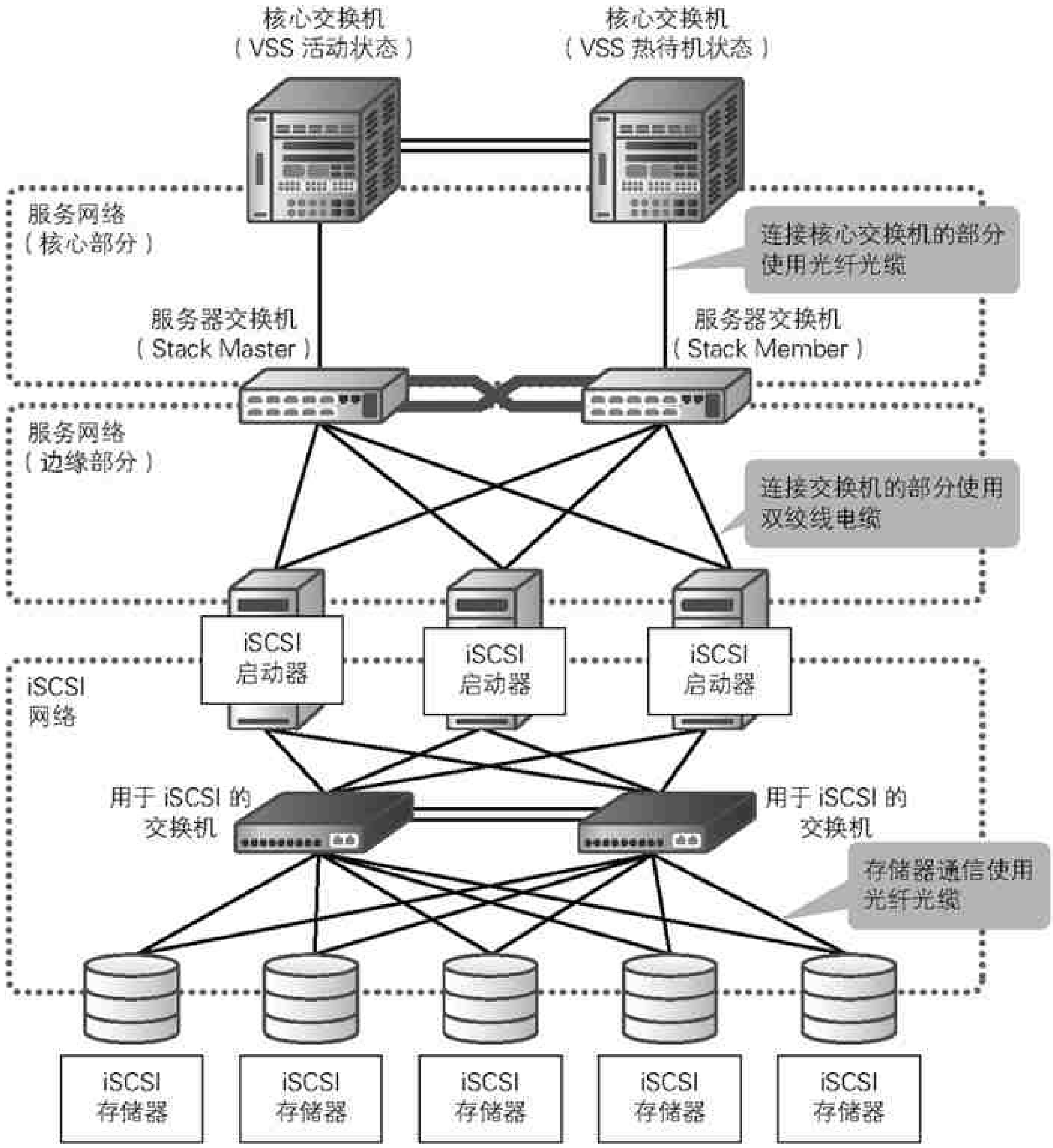


图 **1.2.9**　追求宽频带和高可靠性时选择光纤光缆

1.2.4.3　通过大小分类决定使用哪种双绞线电缆

使用双绞线电缆时，首先要注意它的大分类。双绞线电缆越是接近宽频带的规格就会越容易受到电磁干扰，所以必须慎重选择合适的大分类。例如，对于 1000BASE-T 笔者推荐使用 5e 以上的规格，对于 10GBASE-T 则推荐使用 6A 以上的规格。在设计时我们应明确规定使用哪种大分类，这样以后才不会糊里糊涂、不知所措。

双绞线电缆还有直通线和交叉线这两个小分类，所以使用双绞线电缆时我们还必须决定选择其中哪一类。当 Auto MDI/MDI-X 功能处于激活状态时我们可以不必在意小分类，然而当该功能未被激活时，小分类不相符的话就会无法连通。我们可以立下类似这样的约定：连接同种网络设备时使用交叉线，连接服务器、PC 以及不同种类的网络设备时使用直通线。

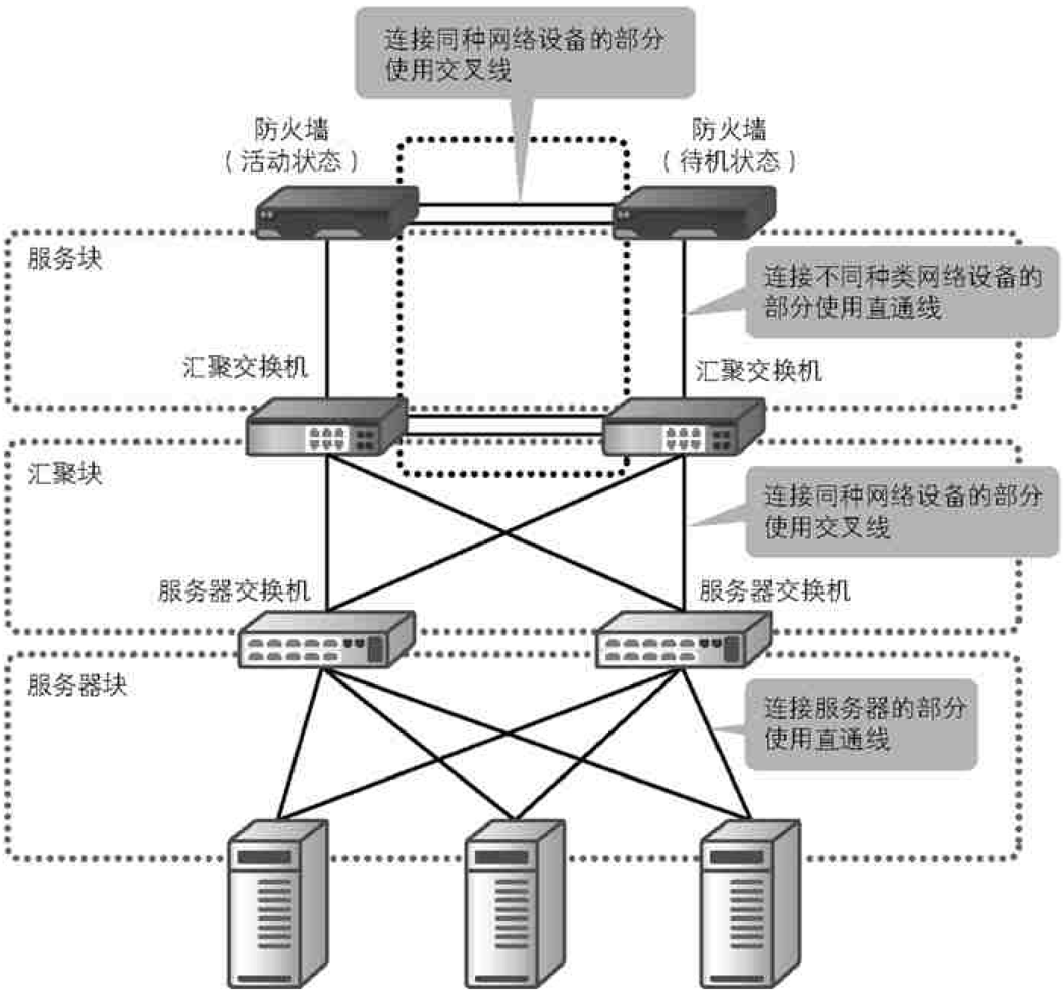


图 **1.2.10**　当 **Auto MDI/MDI-X** 功能未被激活时决定是使用交叉线还是直通线

1.2.4.4　预先决定好使用线缆的颜色

线缆的颜色很重要，这一点容易被人们忽视。预先确定好在什么地方，或者说针对什么机器使用什么颜色的线缆，我们就能一眼看出该线缆的用途，以便进行管理。一般说来，按照线缆的小分类划分颜色较为常

见，例如交叉线用红色、直通线用蓝色。这样，当线缆有问题时我们一看就知道要准备什么线缆去替换，有助于快速解决问题。

这么一说，笔者想起来曾在某个服务器端见过色彩斑斓犹如彩虹的线缆阵容。总共有九种颜色，的确是鲜艳无比、漂亮非凡。然而，对于如此多彩的 LAN 网线我只觉得惊讶不已，这样安排的话，不预先囤积大量的备用线缆，以后扩展系统时得花多少精力去配置它们啊。所以，我们在定义颜色时还是着眼于其中的几个关键点比较好。

到这里我们已经讲述了应该选择怎样的线缆。也许有的读者会想，全都使用光纤光缆不就好了？可能的确如此，如果贵公司财力雄厚可以花钱如流水，那么无可厚非，然而并非所有企业都是如此，光纤的线缆和连接器都是比较贵的，而且有些设备压根就没有光纤连接器。在这个充满艰辛的世界上生存，我们得勤俭节约才行，所以还是多多精打细算，仅在机器间距离遥远以及需要提高传输质量的少数情况下使用光纤光缆吧。

1.2.5　端口的物理设计出乎意料地重要

在这里，我们要设计将什么对象连接到何处的基本规则。内容虽极其单纯，但对今后的运行管理却影响巨大。如果将服务器和 PC 不加考虑地随意连到空闲端口上，发生问题时就会弄不清到底是哪里连接着哪些机器。不统一基本规则就去连接的话，系统结构就无法在脑海中留下清晰的印象。所以我们一定要在设计阶段制定好明确易懂的规则，明确什么设备连在何处。

1.2.5.1　必须统一规划连接到哪里

保持统一性这一点在设计阶段非常重要。因此，如果现有系统中已经有了基本规则，我们最好去遵循它。无视现有的基本规则去进行系统架构和扩展只会让今后的局面混乱不堪。如果是新建一个系统，那就要从无到有地制定出基本规则。这时，设计人员必须要制定出明确易懂、便于扩展的基本规则。

以笔者制定的基本规则为例：如果连接是并列配置和上下配置的网络设备的端口，由于它们不太会发生增减，所以要从尾号端口开始使用；相反，如果连接的是服务器和 PC 的端口，由于它们很可能会发生增减，所以要从小号端口开始使用。我们要像这样针对设备的每项功能制定出

基本规则。

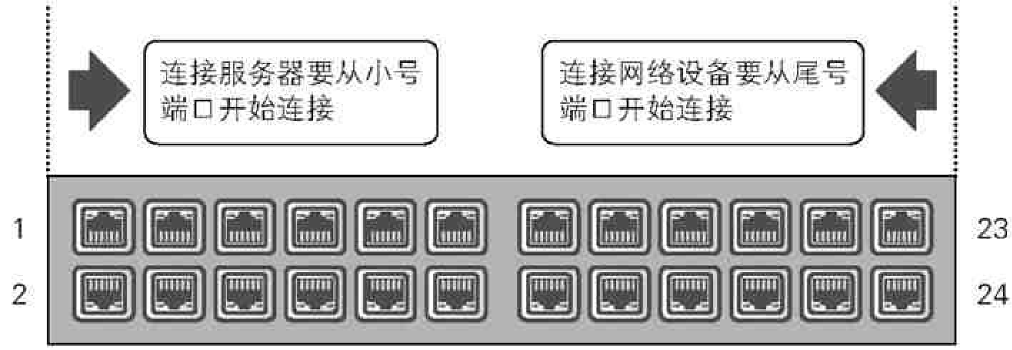


图 **1.2.11**　保持连接的统一性

1.2.5.2　速率和双工、**Auto MDI/MDI-X** 的设置也要统一规划

在前文中已经讲过，当相连的两台设备各自的端口速率和双工不一致时，通信是不成立的，所以我们必须制定好基本规则以确保它们一致。和上一节一样，如果现有系统中已经有了基本规则，我们最好去遵循它。如果是新建一个系统，我们就要基于连接对象去考虑速率和双工的设置。最近，连接双方发生兼容问题的情况已经越来越少，所有端口都是自动设置的情况则越来越多。这固然是一种可能存在的理想状态，不过在这里，将连接双方手动设成固定值还是采用自动设置这件事情并不重要，重要的是制定一个清晰的基本规则以保证两台设备保持步调一致。

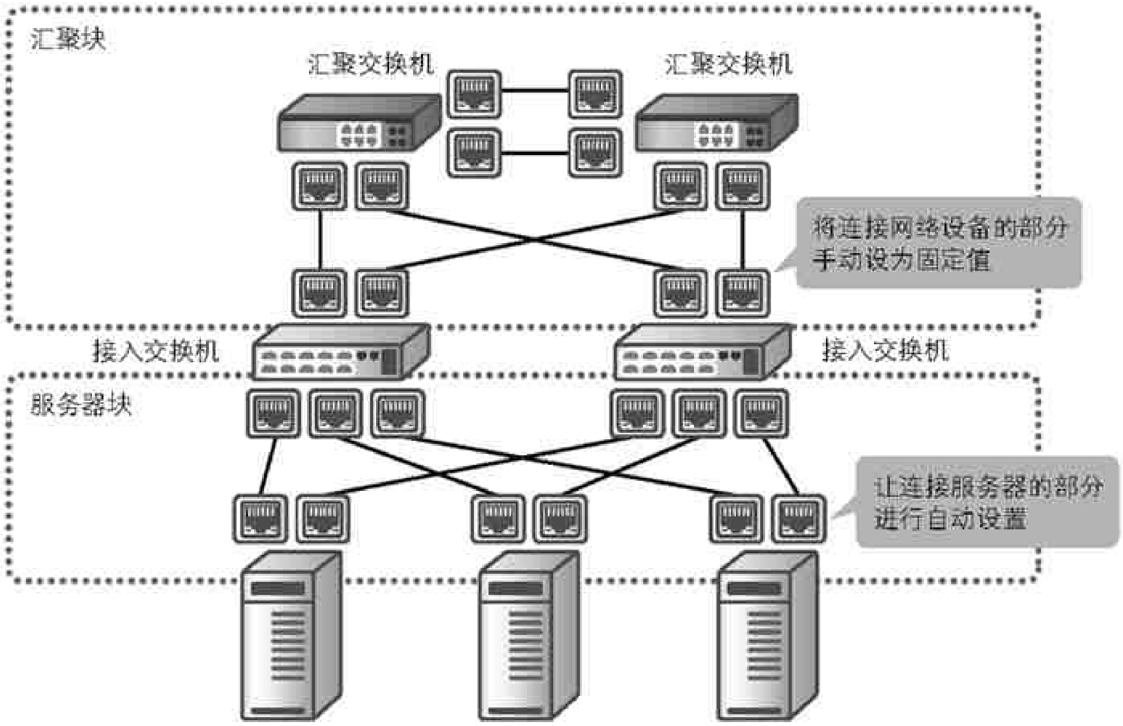


图 **1.2.12**　让两台设备的速率和双工保持一致非常重要

1.2.6　巧妙地配置设备

将设备分别配置到楼层的何处，这也是物理设计的内容之一。我们要在考虑整个物理结构及其可扩展性的基础上将设备分配到楼层各处。此外，如何将设备配置到机架中去也非常重要，我们应熟悉设备规格并仔细斟酌将它们配置到哪里以及具体该如何配置。

1.2.6.1　将核心交换机和汇聚交换机置于中央部位

我们应将设备置于楼层的何处，又该如何进行具体配置呢？这是个关系到网络的物理结构和可扩展性的问题。大多系统都是按照设备的角色和功能去安排分层结构中的网络构造的，按照排列层次去考虑设备的配置应该比较高效。

举个例子，假设该系统由核心交换机[9]、汇聚交换机[10]和接入交换机[11]这三种要素构成。这种情况下，接入交换机之外的其他设备是不会在突然之间大量增设的，于是我们可以将这些设备配置在楼层和机箱室等指定场地的中央部位。而接入交换机可能因服务器台数的增加而增加，我们应将它们分散配置以应不时之需。接入交换机有端列头（End of

Row）和机架顶（Top of Rack）两种配置类型，各自也是优缺点并存，我们在选择时应和管理成本和运行的人员进行充分的探讨。

> [9] 核心交换机是在系统中发挥核心作用的交换机，将汇聚（分布）交换机汇集到一起。

> [10] 汇聚交换机是将接入交换机汇集到一起的交换机，又叫分布交换机。

> [11] 接入交换机是连接服务器的交换机。

表 **1.2.3**　交换机有两种配置类型

比较事项	端列头式	机架顶式
接入交换器的设置单位	整个机架列	整个机架
管理台数	少	多
每台接入交换机连接的服务器数量	多	少
跨机架的线缆	多	少
线缆成本	低	高
可扩展性	低	高
灵活性	低	高

采用端列头式配置可以一气呵成地完成设备连接

端列头式配置是以整个机架列为单位配置接入交换机的类型。由较大的模块型交换机接纳较多的服务器，如果服务器增加，就增设接口模块来对应。采用这种配置类型会有大量的线缆在机架之间穿插交织，线缆的

布线作业比较繁琐，但是需要管理的接入交换机的数量比较少。

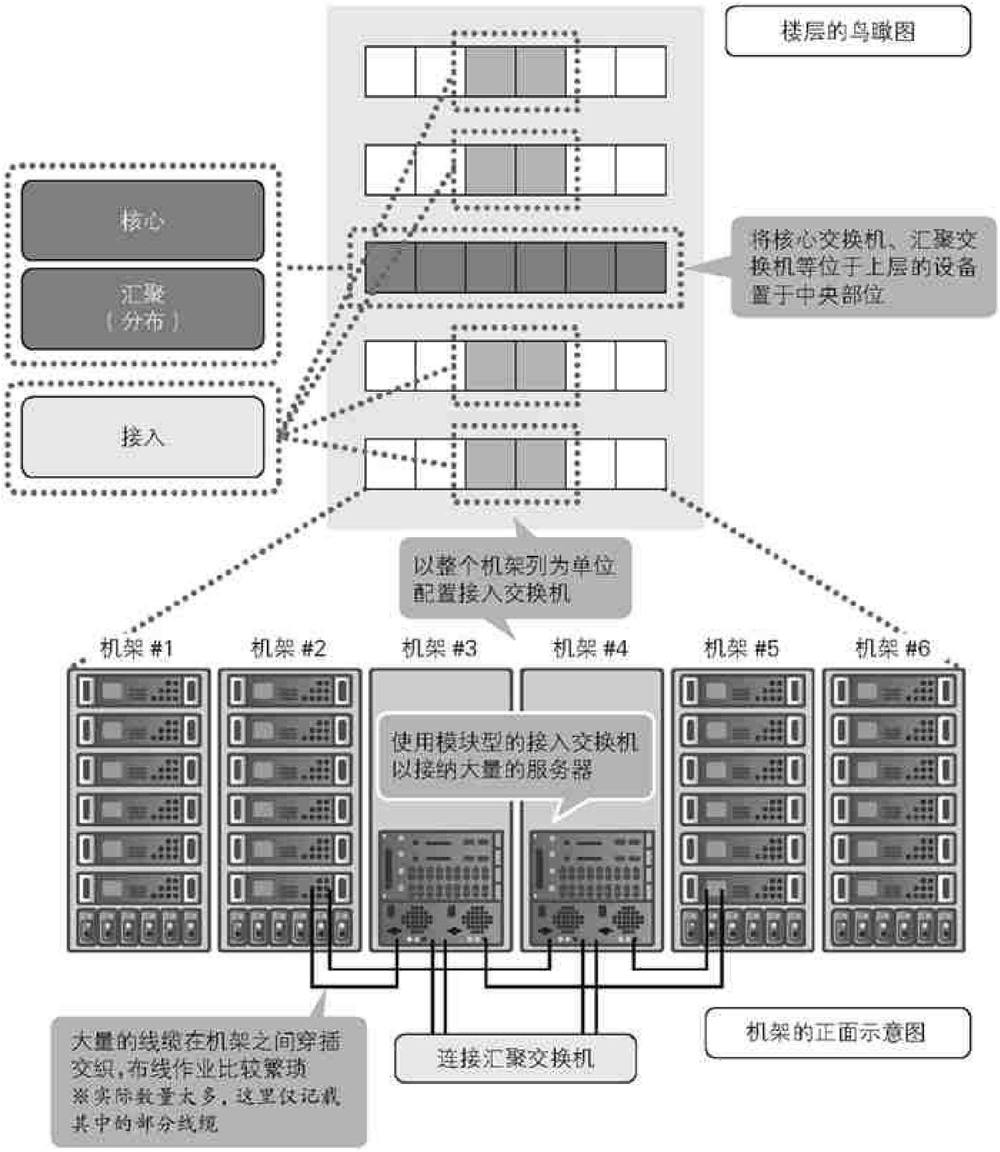


图 **1.2.13**　采用端列头式配置时只需管理少量设备即可

采用机架顶式配置可以在整个机架之间进行连接

机架顶式配置是以整个机架为单位配置接入交换机的类型。由较小的固定型交换机接纳机架内的所有服务器，机架内的服务器配线无需太长，在机架内部能连接到设备即可。由于是在一个个机架中配置接入交换机，需要管理的设备数量会比较多，不过线缆的布线作业比较轻松，线缆成本也有所降低。

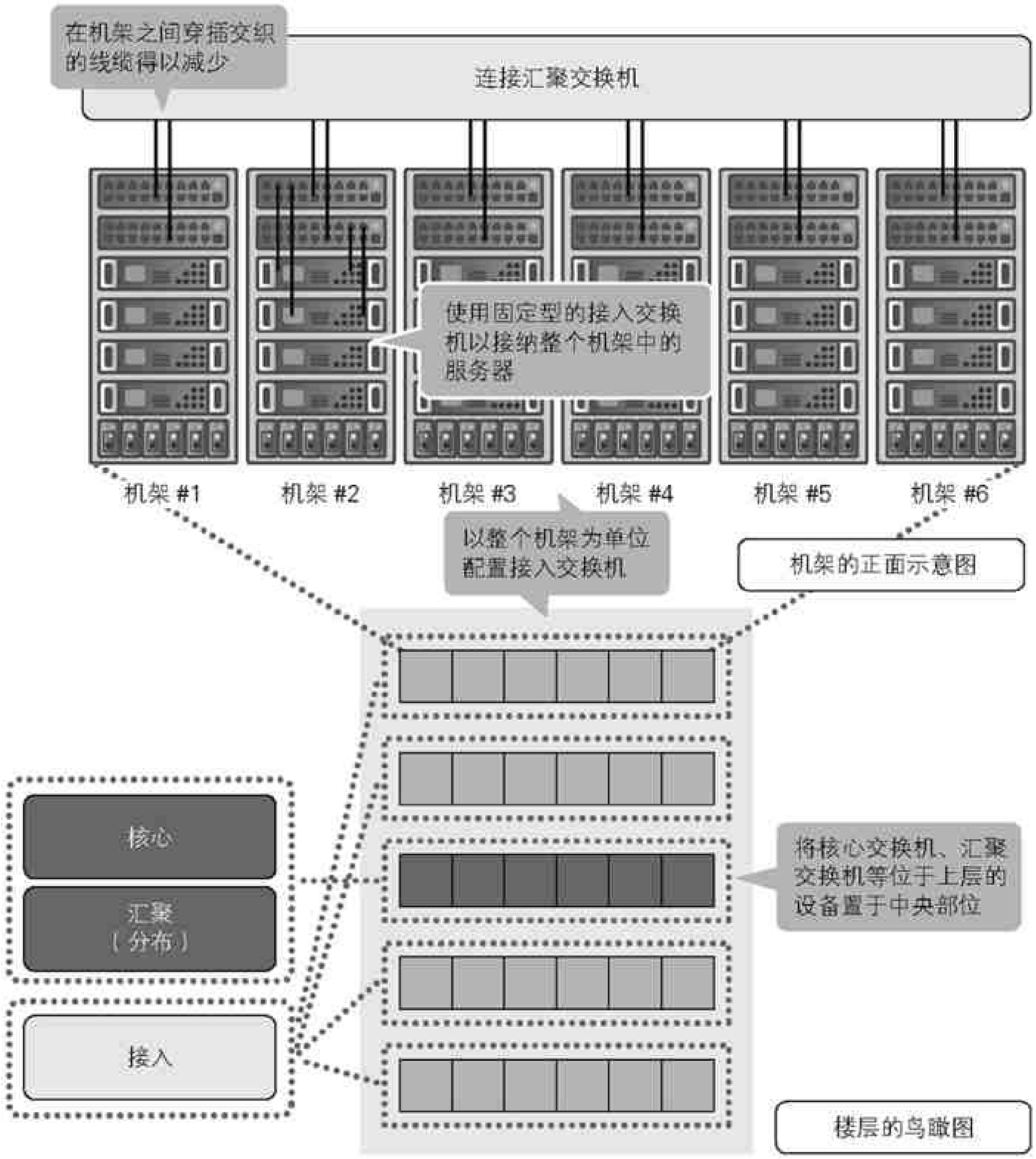


图 **1.2.14**　采用机架顶式配置时线缆的布线作业比较轻松

1.2.6.2　要考虑设备中空气吸入和排出的方向

将设备置于机架的什么地方，具体又是如何安装的，这也是个重要的问题。最近，也有数据中心对空气的流动进行了细致的设计，通过向机架通道交替送出热空气和冷空气来提高整个楼层的气冷效率。在这种空气调节的设计中，保证冷热空气不互相混合非常重要。如果我们不了解各种设备在工作时哪面会吸入空气哪面又会排出空气，就会导致冷热混流。所以，一定要在仔细了解设备规格之后再将它们安装到机架上去。

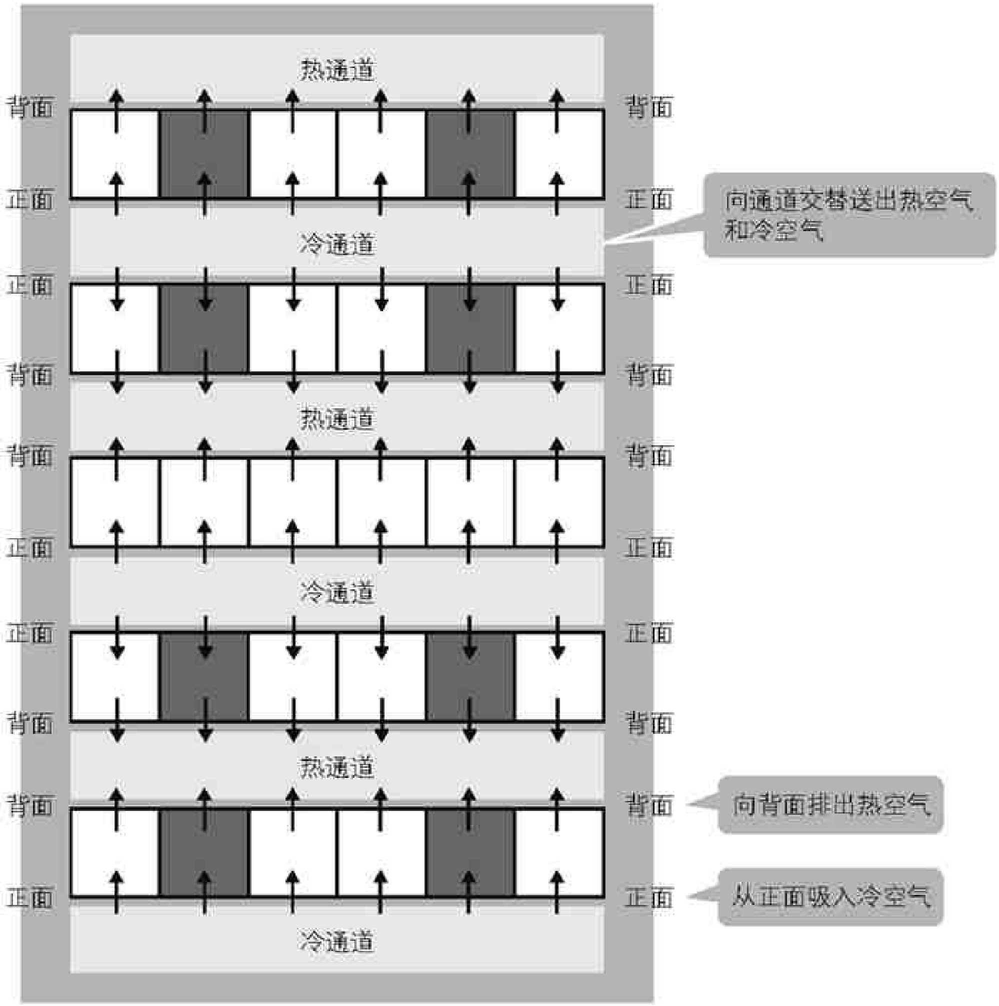


图 **1.2.15**　充分考虑设备吸入和排出空气的方向后再将设备安装到机架上

1.2.6.3　从两套系统获取电源

网络设备是非常精密的电子机器。无论我们对设备的功能和物理结构做了怎样的冗余配置，没有电的话设备就无法工作。所以我们还要对电源进行设计，确保当系统某处发生断电情况时也能够继续提供服务。

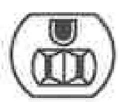
切莫弄错电源插头

用于网络设备和服务器的电源种类非常繁多，我们应根据设备的适用电流（A）和电压（V）去选择引入机架的电源。这里必须注意的是电源插座的形状，电源插头和电源插座的形状不吻合的话就会因插不进去而无法通电，这是不言自明的。因此我们务必多加注意，不要到时候在机

架前束手无策，只有痛哭流涕的份儿。

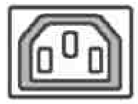
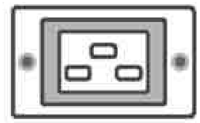
常用的电源插座有四种形状。如果电源电压为 100V，那么可以使用和家用电源插座一样的 NEMA 5-15 型或可锁式的 NEMA L5-30 型插座；如果电源电压为 200V，那么可以使用 NEMA L6-20 型或 NEMA L6-30 型插座。L 表示能否咔哒一声就锁住，L 后面的数字 5 代表 100V，6 代表 200V，横杠后面的数字代表安培数。我们应在考虑设备适用的电压和整个机架的安培数之后选择引入哪种电源。模块型网络设备和刀片服务器的安培数会因其安装的模块和刀片的数量及种类而变化。我们在计算安培数的时候要为可扩展性考虑，保留一定余地，避免出现机架上虽然有空的插槽却不能使用的窘况。

表 **1.2.4**　电源插座有多种形状

事项	NEMA 5-15	NEMA L5-30	NEMA L6-20	NEMA L6-30
连接器的形状				
电压类型	100V	100V	200V	200V
电流	15A	30A	20A	30A
是否可锁	×	○	○	○

电源引入机架之后会出现两个分支，一个是 PDU（Power Distribution Unit，电源分配单元 / 电源板），另一个是 UPS（Uninterruptible Power Supply，不间断电源）。如果电压为 100V，那么插座形状和家用插座是一样的，直接使用即可。但如果电压为 200V，我们就要注意区别 PDU 和 UPS 了，它们的插座形状分别为 IEC320 C13 型和 IEC320 C19 型，要使用符合它们形状的电源线缆才行。

表 **1.2.5**　电压为 **200V** 时应注意 **PDU** 和 **UPS** 的插座形状

连接的插座	IEC320 C13	IEC320 C19
插座的形状		
用途	连接 200V 的 PDU	连接 200V 的 UPS

按照用途区别使用电源系统

要想对电源进行冗余配置，就必须具备一个前提，那就是在一个机架中布设两套电源系统，这是绝对条件。如果仅布设了一套电源系统，那么无论在机架内做怎样的电源冗余配置都是毫无意义的。这里，我们以机架中布设有 A、B 两套电源系统为前提进行说明。根据实际需要，我们可能会在两套系统中都设置 UPS 或仅对其中一套设置 UPS。

对于电源单元已做冗余配置的机器，我们无需考虑太多，让其从 A、B 系统分别获取电源即可。这样，即使其中一套电源发生故障，服务也不会受到丝毫的影响。然而对于电源单元未做冗余配置的机器，我们就要根据设备本身的冗余状态（活动 / 待机、Stack Master/Member 等）和连接路线将电源获取系统分开了。如果活动机和备用机都是从同一系统获取电源，那么该系统发生故障时我们就会一筹莫展、无计可施。所以我们应该将电源系统分开使用，让活动机从 A 系统获取电源，让备用机从 B 系统获取电源。

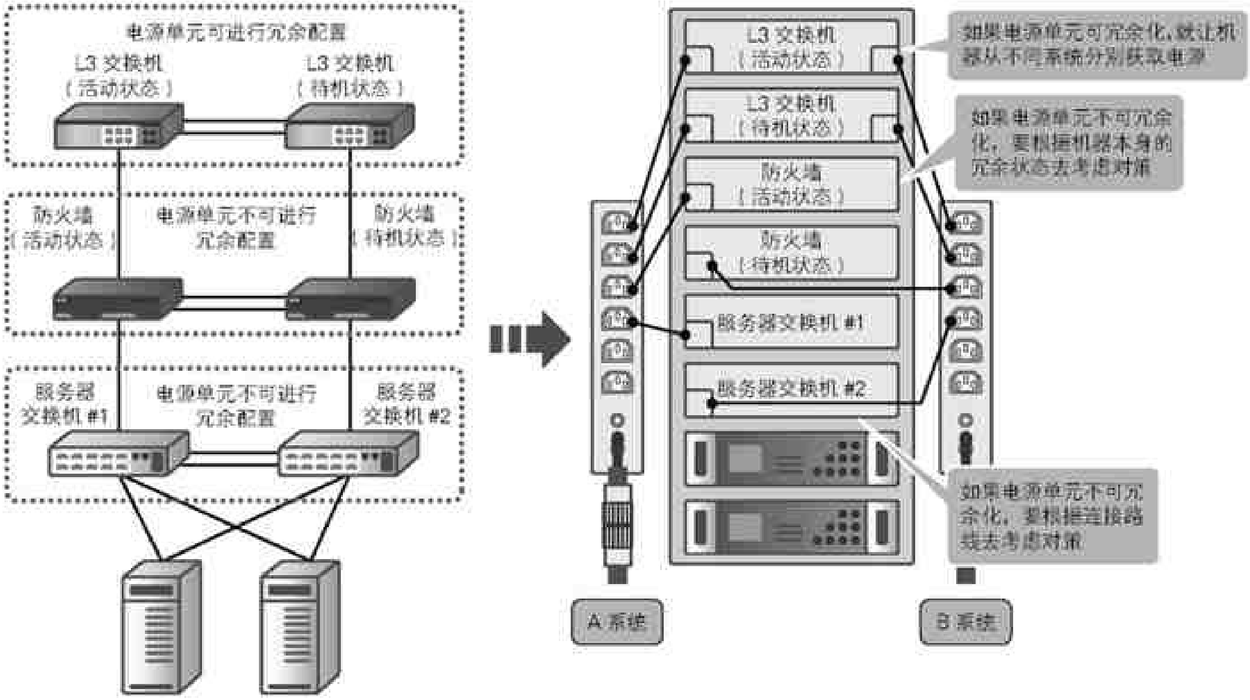


图 **1.2.16**　划分系统获取电源

1.2.6.4　切莫超过最大承重

大多数数据中心和机房的地板都是双层构造，便于从下方吹送冷空气和布设机架之间的线缆。因此，为防止机架下沉我们还需确定最大承重值。最大承重分两种，分别是机架的最大承重和地板的最大承重。机架的最大承重指的是机架所能承载的重量，地板的最大承重则指地板每平方米单位所能承载的重量。如果在空处配置了太多服务器和网络设备，机架无疑会下沉。所以我们在考虑设备配置时还必须了解它们的最大重量。模块型网络设备和刀片服务器的重量会因其安装的模块和刀片的数量而变化，我们要算出安装了各种设备时的最大总重量，并为可扩展性考虑，留出一定的余地。

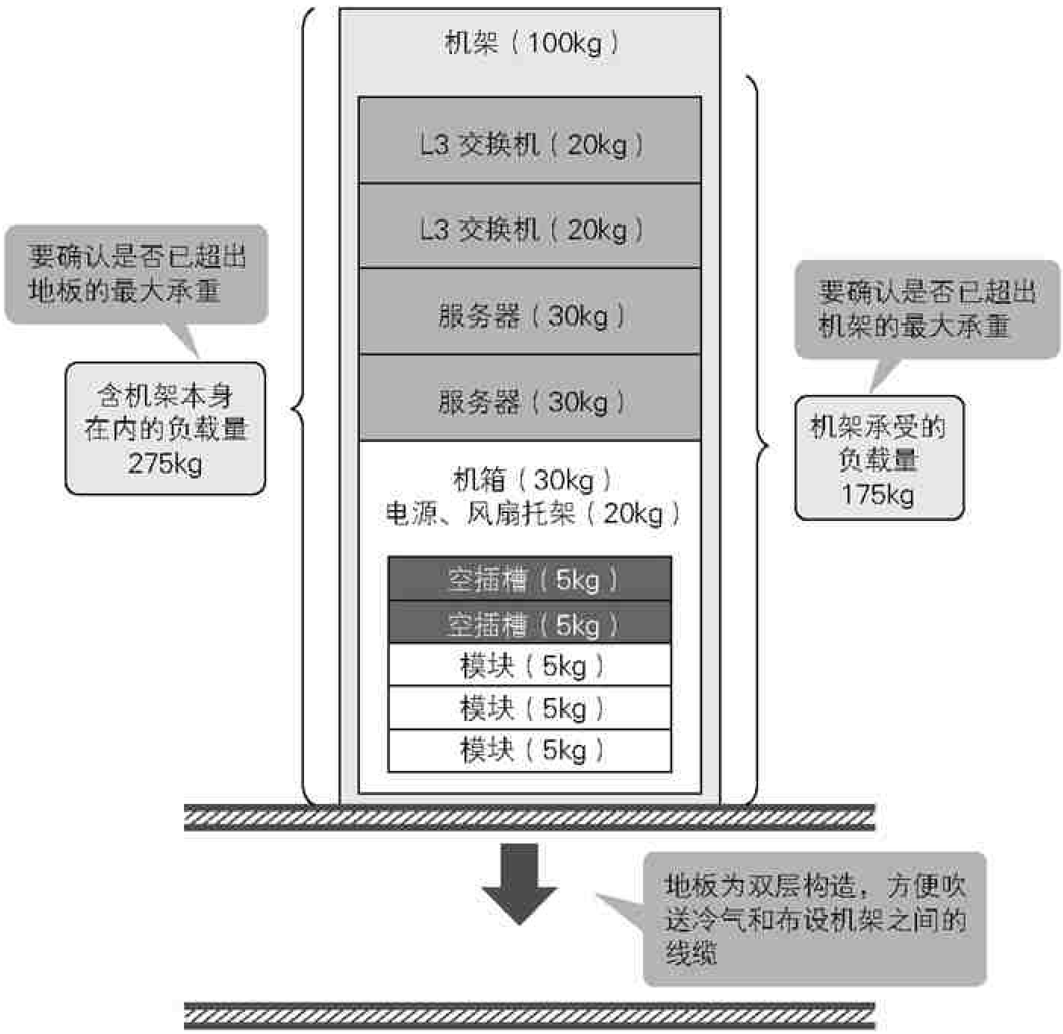


图 **1.2.17**　切莫超过最大承重（实际情况可不会像图例中那么轻）

第 **2** 章　逻辑设计

本章概要

本章主要介绍用于服务器端的数据链路层和网络层的技术，以及使用这些技术时的设计要点。

最近这几年，和网络相关的基础技术并未取得较大发展，不过倒是变得越发纯粹和简练了。然而，用于服务器端和客户端的刀片服务器和虚拟化技术等却在日新月异地进步，人们追求的网络形式也在不断发生变化。我们要好好理解这些技术和规格，设计出最符合客户需求的逻辑结构，以灵活应对这些变化。

2.1　数据链路层的技术

数据链路层在物理层的上方提供了一种能够正确且稳定地传输比特流的结构。物理层仅仅负责将比特流转换成信号并传递给线缆，既不知道信号会发往何处，也不在意某一位是否会在某处丢失。数据链路层则刚好能够弥补物理层的这个短处，它能生成通往相邻设备的逻辑传输路径（数据链路），检查出其中的传输错误并将错误修复，从而确保物理层的可靠性。

2.1.1　数据链路层是物理层的帮手

数据链路层是用于确保物理层可靠性的层。数据链路层中的“数据链路”表示由它生成的、和相邻设备（节点）之间的逻辑传输路径。

为了判断应针对哪个节点生成数据链路以及生成的数据链路中是否有位缺失，数据链路层会对数据进行封装处理，确保物理层的可靠性。在数据链路层进行的这种数据封装处理叫作成帧，封装之后的数据叫作帧。在数据链路层定义了成帧的各种方式。

数据链路层位于网络层和物理层之间，是从下往上数的第二层。发送信息时，数据链路层将来自网络层的数据（数据包）封装成帧后交给物理层；接收信息时则对来自物理层的比特流做一个和成帧恰好相反的处理，然后交给网络层。

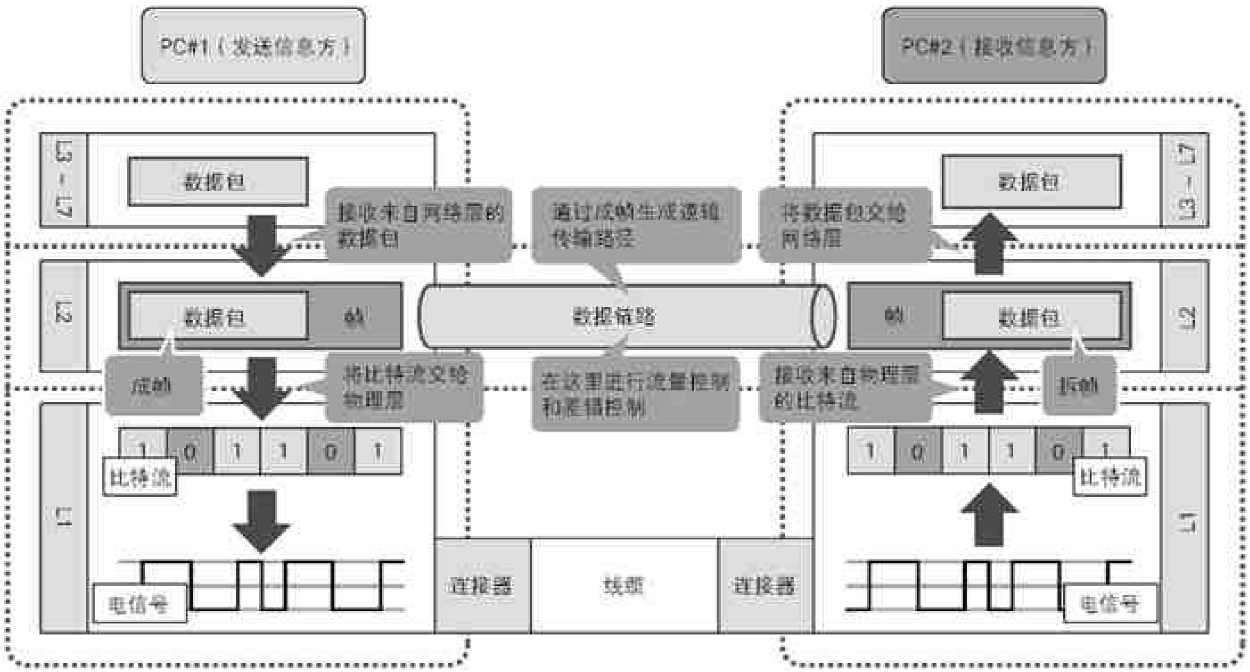


图 **2.1.1**　在数据链路层将数据封装成帧

用以太网标准进行成帧处理

下面，我们来看看承担着数据链路层核心作用的封装处理“成帧”具体是怎么一回事。

数据链路层和物理层是同甘共苦、休戚与共的关系，成帧的规格也和二者紧密相连。以往有令牌环网、帧中继、PPP 等林林总总的各式规格，不过现在我们只需记住以太网（IEEE802.3）这一个就够了，接下来本书将为你讲解以太网（IEEE802.3）是如何进行成帧处理的。

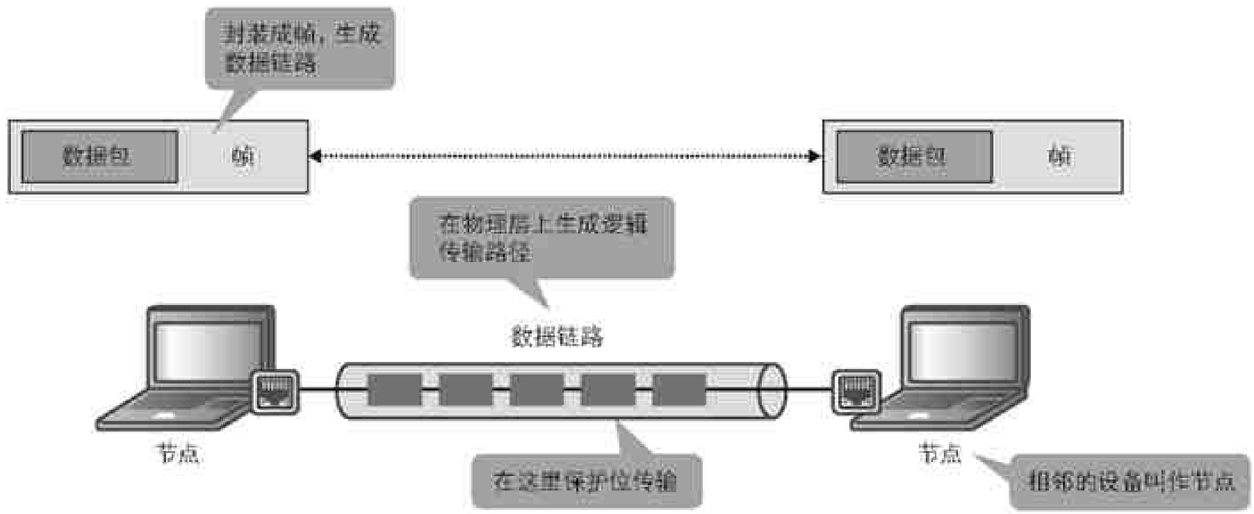


图 **2.1.2**　通过封装成帧确保位不会丢失

以太网的成帧规格可粗略划分为 Ethernet Ⅱ（DIX）和 IEEE802.3 这两种。IEEE802.3 是将 Ethernet Ⅱ 改良之后根据 IEEE 标准制定出来的，但实际上大多数数据通信使用的还是 Ethernet Ⅱ规格。Ethernet Ⅱ的帧格式已经保持 20 多年没变了，是一种非常简约而又容易理解的格式。Ethernet Ⅱ给数据（数据包）添加前导码、帧头和 FCS（Frame Check Sequence，帧校验序列）。

前导码相当于发送帧的信号

前导码是一个 8 字节（64 位）的位数组，相当于一个“我要发送帧了哦”的信号。它和“10101010……（中间省略）……10101011”的位数组一定是一样的。对方看到这个特别的位数组之后就能知道即将会接收到帧。

帧头决定将信息发至何处

帧头由三个字段构成，分别为目的 MAC 地址、源 MAC 地址和类型。

- 目的 **MAC** 地址 **/** 源 **MAC** 地址

  那么，发送数据时应生成一个始于何处终于何处的数据链路，又要发送怎样的数据呢？这些都由帧头定义。在以太网中，一般会用一个长度为 6 字节（48 位）、叫作 MAC 地址的识别信息去识别节点。发送数据的节点在目的 MAC 地址里写入目的节点的 MAC 地

址，这便是数据链路的终点；在源 MAC 地址里写入本机 MAC 地址，这便是数据链路的起点。

- 类型

  类型决定数据部分的数据包使用何种协议，网络层协议决定这里的具体数值是多少。下表中列出了部分具有代表性的协议类型代码。

表 **2.1.1**　具有代表性的协议类型代码

类型代码（十六进制）	协议
0x0800	IPv4（Internet Protocol Version 4，网际协议第四版）
0x0806	ARP（Address Resolution Protocol，地址解析协议）
0x86DD	IPv6（Internet Protocol Version 6，网际协议第六版）
0x8100	IEEE802.1Q（Tagged VLAN，加标签的虚拟局域网）

数据即 **IP** 数据包

这里说的数据指的是从网络层发来的数据包。有时候也可将其视为 LLC（Logical Link Control，逻辑链路控制）数据，不过 Ethernet Ⅱ中并没有 LLC 子层的处理。因此，我们可以认为数据即 IP 数据包。帧能够传输的数据长度默认最大值为 1500 字节 [1]，这个最大值叫作 MTU（Maximum Transmission Unit，最大传输单元）。当数据长度超过 1500 字节时，必须对整个 MTU 进行数据分段和封装，然后再转发到节点。关于 MTU 的内容将在 3.1.1.3 节中详细说明。

> [1] 我们也可将默认最大值设置成大于 1500 字节的数字。默认最大值大于 1500 字节的帧叫作巨型帧。

用 **FCS** 检查数据错误

FCS（Frame Check Sequence，帧校验序列）字段用来检查数据是否发生了错误。发送方在发送信息时会对帧头和数据进行一定的计算（校验和计算，又称 CRC，即循环冗余校验），并将计算结果添加到 FCS 中去。接收方在接收信息时会再次进行同样的计算，得出的值如果和 FCS 中添加的值一致，则认为帧是正确无误的。如果不一致，则会认为数据在传输过程中发生了错误，于是将数据丢弃。FCS 就是这样在数据链路中起着错误控制的作用。

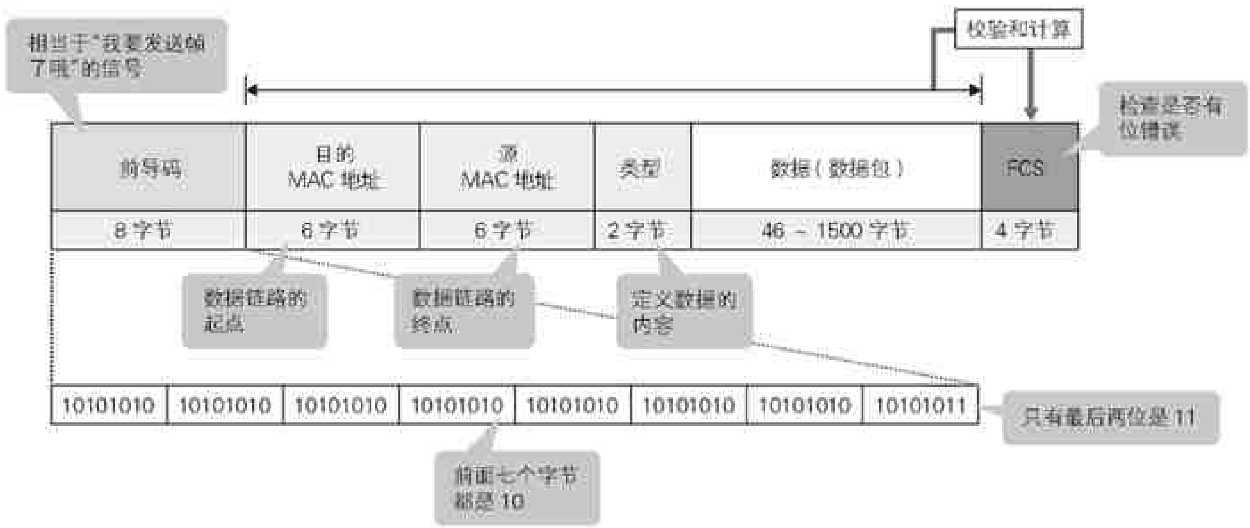


图 **2.1.3**　以太网的帧格式非常清晰，容易理解

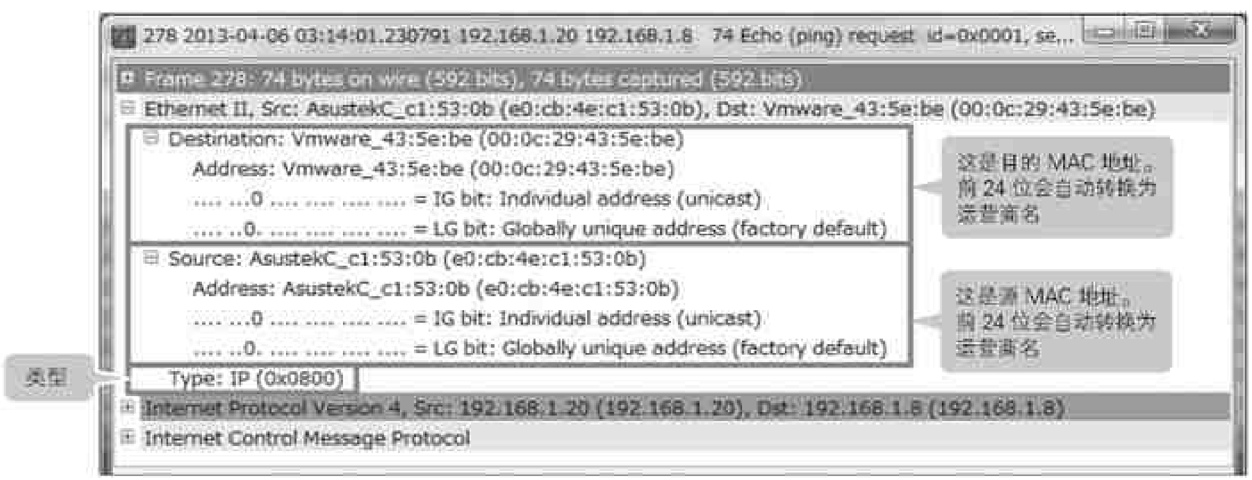


图 **2.1.4**　用 **Wireshark** 分析以太网帧的画面

※ 用 Wireshark 进行分析时无法显示出前导码和 FCS，这是因为它们在 Wireshark 接收信息之前就已经被卸掉了，分析画面中仅显示除它们之外的那些信息（帧头和数据）。

看前 **24** 位就能知道网卡的运营商

MAC 地址是由 48 位构成的独一无二的识别信息，在以太网中扮演着数据链路起点和终点的角色。它的具体写法如 E0-CB-4E-C1-53-CB、00:0c:29:43:5e:be，每 8 位就用一个连字符或冒号隔开，用十六进制表示。

MAC 地址的前 24 位和后 24 位有着不同的含义。前 24 位是由 IEEE 管理的独一无二的运营商编码，叫作 OUI（Organizationally Unique Identifier，组织唯一标识符）。看这部分我们就能知道通信节点的网卡是哪家运营商提供的。顺便提一句，下面的网页中有 OUI 的公开信息，排除故障时不妨参考一下。

URL http://standards.ieee.org/develop/regauth/oui/oui.txt （截至 2014 年 1 月的信息）

另外，预先在 Wireshark 的 Capture Options → Name Resolution 中勾选 Enable MAC name resolution 选项，MAC 地址的运营商编码就能自动转换成运营商名。这个功能意想不到地方便，笔者经常使用。

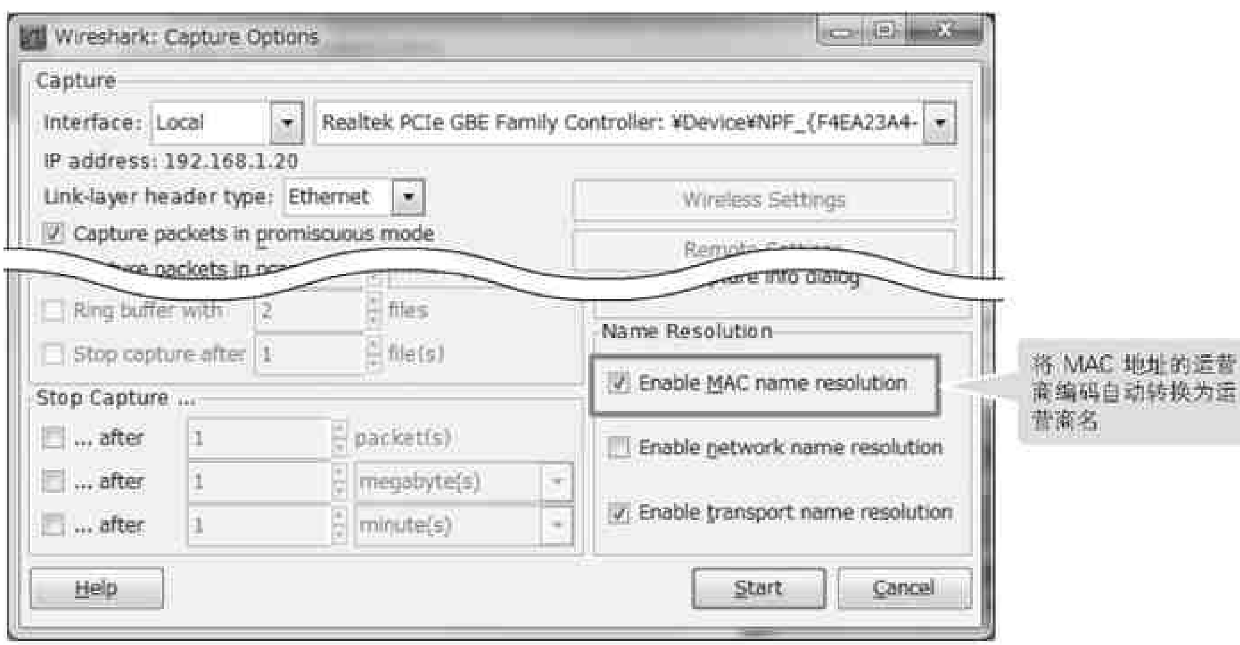


图 **2.1.5　Wireshark** 可将运营商编码自动转换成运营商名

后 24 位是运营商内部管理的独一无二的编码。由 IEEE 管理的独一无二

的前 24 位和由运营商管理的独一无二的后 24 位，这两个要素使 MAC 地址成为了世界上独一无二的地址。

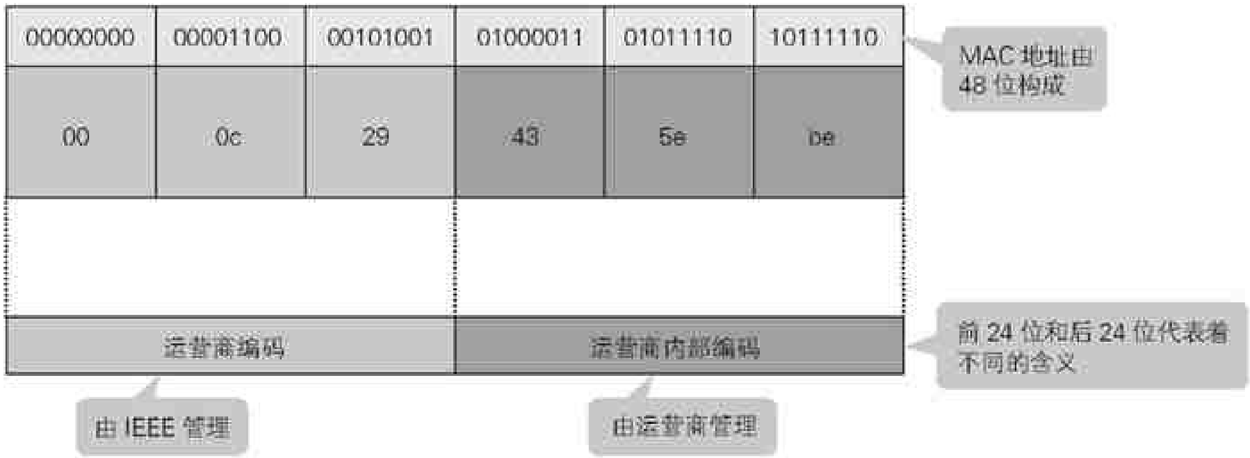


图 **2.1.6　MAC** 地址中的每段 **24** 位都代表着不同的意思

特殊的 **MAC** 地址

以太网中的所有通信都是一对一的形式吗？并不一定。一般情况是这样的：所有节点共享一根线缆（传输媒介），同时生成大量的逻辑传输路径（数据链路）与众多的对象互通信息。以太网按通信方式划分为单播、广播和多播三种帧类型，比较常用的是单播和广播。

- 单播

  单播是一对一的通信，这应该很容易理解吧。MAC 地址基本上都是独一无二的，因此对应某个节点可以只生成一条数据链路，源 MAC 地址和目的 MAC 地址指的就是这两个节点的 MAC 地址。在现代以太网的环境中这种通信所占的比例很大，电子邮件和互联网的通信就属于这种通信类型。

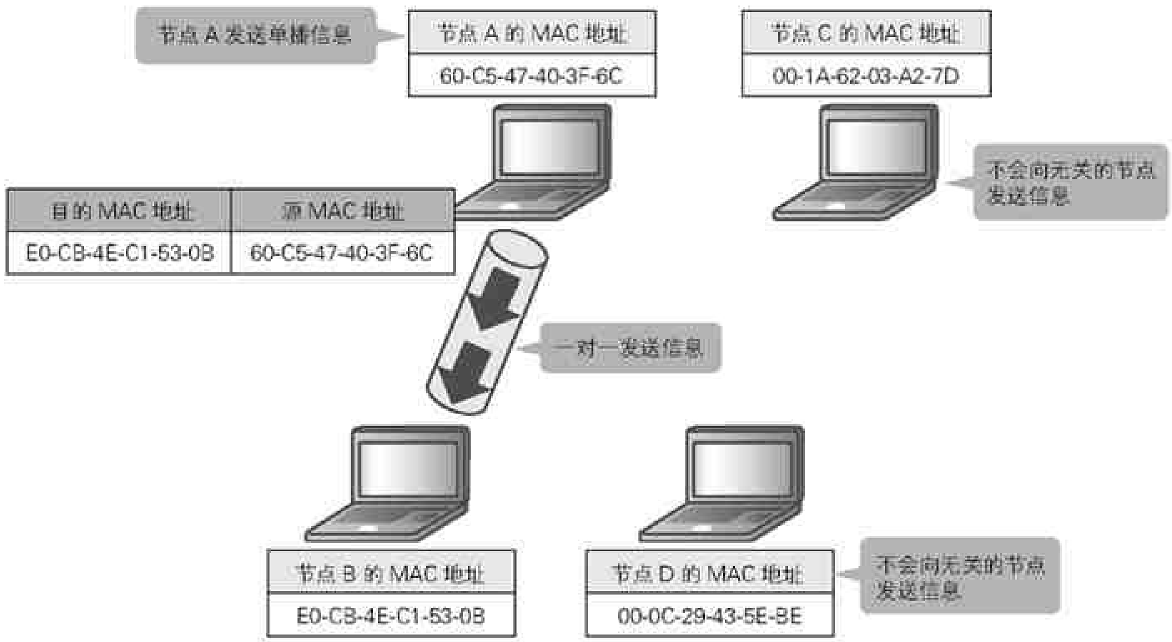


图 **2.1.7**　单播是一对一的通信

- 广播

广播是一对多的通信，这里所说的“多”是指同一个网络中的所有节点。如果某个节点发送了广播信息，那么所有的节点都会收到该信息。广播能够波及的范围叫作广播域。广播的源 MAC 地址就是发送节点的 MAC 地址，目的 MAC 地址则比较特殊，稍微有点不同，写出来是 FF-FF-FF-FF-FF-FF 的形式，用位表示的话全部都是 1。

举一个比较典型的广播通信的例子，那就是 ARP（Address Resolution Protocol，地址解析协议）。我们来看下 ARP 大致的工作原理。假设节点 A 要对节点 B 进行单播，可是节点 A 并不知道节点 B 的 MAC 地址，为了弄清这个地址节点 A 就要用到广播，通过广播向所有节点发问“请告诉我节点 B 的 MAC 地址是多少”，弄清了之后再和节点 B 进行单播通信。关于 ARP 的内容将在 2.1.3 节中详细说明。

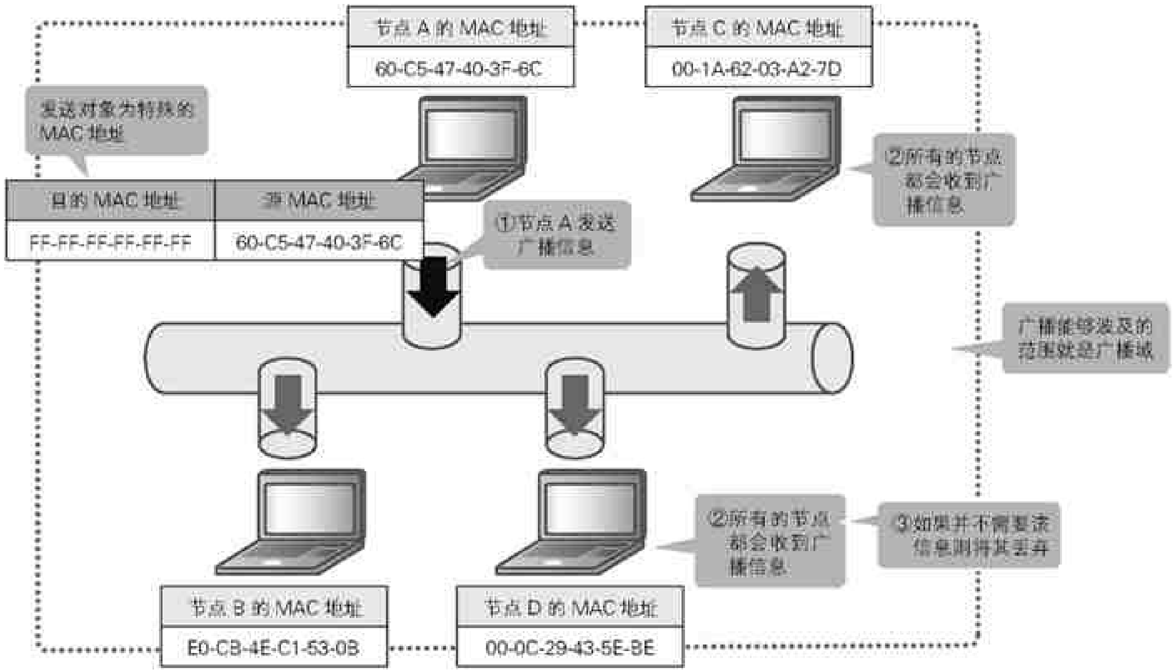


图 **2.1.8**　广播是针对所有节点一气呵成地发送信息

- 多播

多播是一对多的通信，看起来似乎和广播差不多，但这里的“多”指的是特定小组（多播组）中的节点。如果某个节点发送了多播信息，那么这个小组中的所有节点都会收到该信息。多播的源 MAC 地址就是发送节点的 MAC 地址，目的 MAC 地址则比较特殊，稍微有点不同。多播 MAC 地址前面第 8 位的 I/G（Individual/Group）位是 1。用于广播的 MAC 地址（FF-FF-FF-FF-FF-FF）也算作多播 MAC 地址的一部分。如果要对应多播 IPv4 地址，那么前 25 位是“0000 0001 0000 0000 1001 1110 0”。如果换算成十六进制，那么就是在“01-00-5E”后面的一位加个“0”。“01-00-5E”是管理互联网上全球 IP 地址的 ICANN 所拥有的运营商编码。后 23 位是将多播 IP 地址（224.0.0.0 ～ 239.255.255.255）从后往前数的 23 位复制了一下。多播用于视频发布和证券交易所的应用程序。使用广播时，所有节点都会被强制性地收到信息。相比之下，使用多播时只有启动了该应用程序的节点才会收到信息，信息流动量的效率更高。

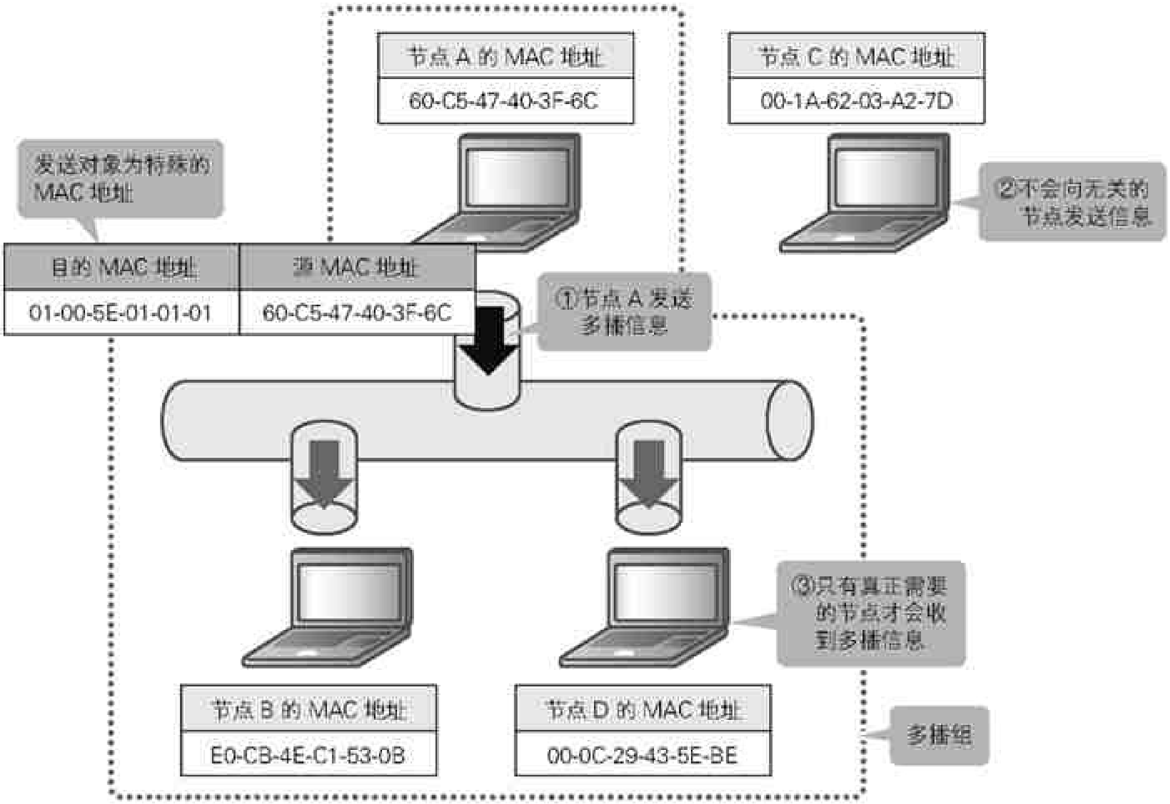


图 **2.1.9**　多播是针对特定的组对象发送信息

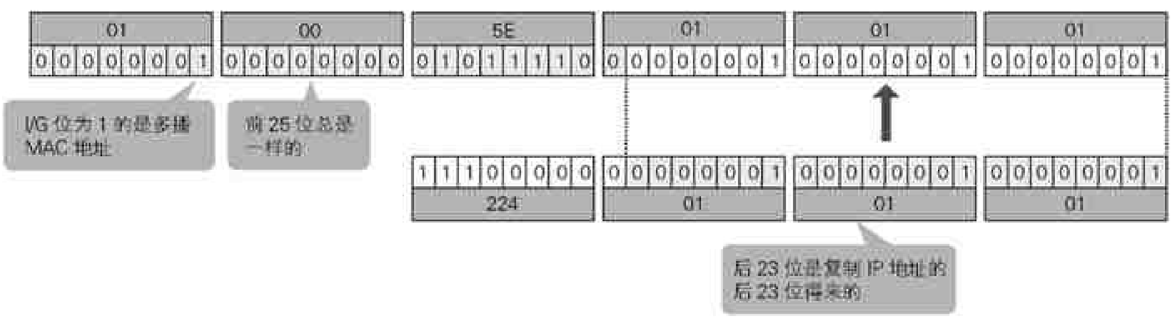


图 **2.1.10**　**IPv4** 多播的 **MAC** 地址是在 **IP** 地址的基础上生成的

2.1.2　数据链路层的关键在于 **L2** 交换机的运作

在数据链路层运行的设备种类繁多，不过我们只要掌握了 L2 交换机，基本上就能应对所有的网络环境了。“交换机”在网络设备的世界里是个常用术语，○○交换机的○○部分代表 OSI 参考模型的层，表示它们是根据哪一层的信息去切换转发地址的。转发地址的切换叫作“交换”，L2 交换机根据数据链路层（L2）的信息，也就是 MAC 地址的信息对帧进

行 L2 交换。

表 **2.1.2**　有各种各样的交换

层		交换机	根据什么进行交换
L5～L7	会话层 ～ 应用层	L7 交换机（负载均衡器）	应用程序
L4	传输层	L4 交换机（负载均衡器）	TCP/UDP 连接
L3	网络层	L3 交换机	IP 地址
L2	数据链路层	L2 交换机	MAC 地址

2.1.2.1　交换 **MAC** 地址

接下来，我们来看看 L2 交换机是如何进行 L2 交换的。

L2 交换机是根据内存中的 MAC 地址表对帧进行交换的。MAC 地址表由端口和源 MAC 地址的信息构成，看地址表就能知道哪个节点连接着哪个端口。L2 交换机主要有三项职责——登记收到的帧的端口和源 MAC 地址、将来路不明的 MAC 地址进行泛洪处理、删除不再需要的信息。

利用 **MAC** 地址表进行交换

假设节 A 和节点 B 是双向通信的[2]，我们来看看 MAC 地址表在这里是如何派上用场的。

> [2] 这里的假设有一个前提，那就是节点 A 已知节点 B 的 MAC 地址。在实际情况中节点 A 一般并不知道节点 B 的 MAC 地址，所以节点 A 要先通过 ARP 广播确定节点 B 的 MAC 地址之后才能开始通信。

1 →　节点 A 将数据包封装成帧并传递给线缆，目的是发给节点 B。这部分还只是在进行单播发送，源 MAC 地址是节点 A 的 MAC 地址，目的 MAC 地址是节点 B 的 MAC 地址。

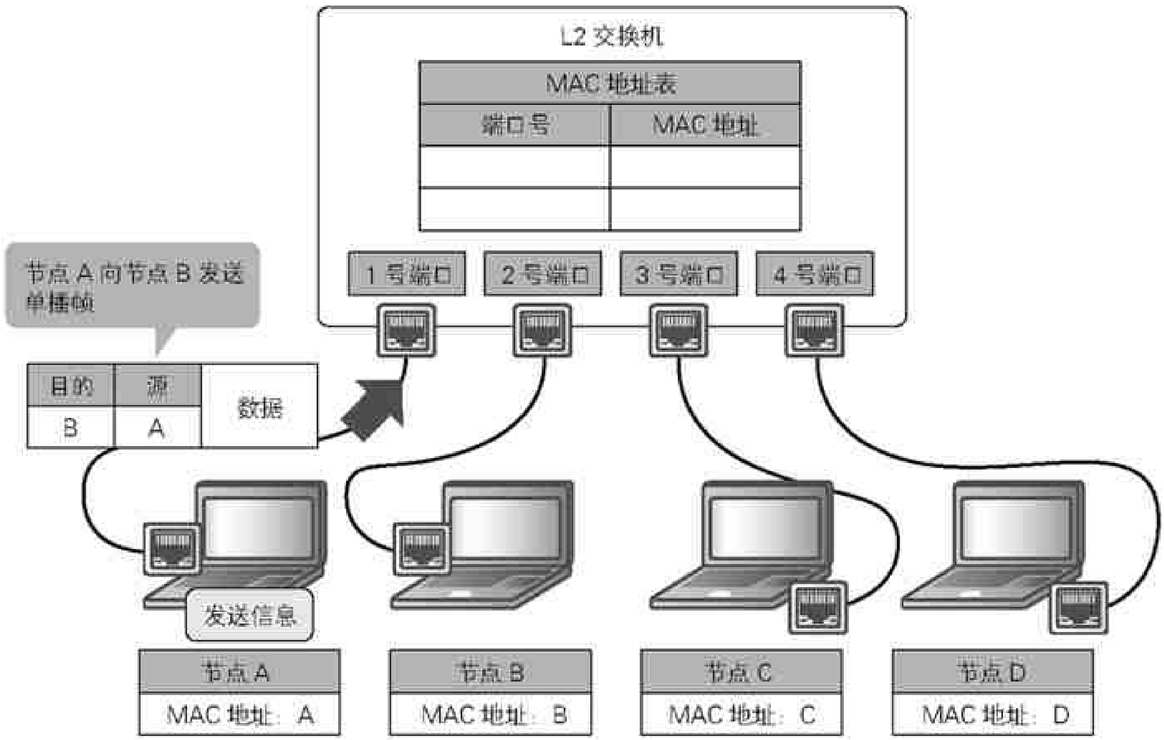


图 **2.1.11**　节点 **A** 向节点 **B** 发送单播帧

2 →　交换机收到帧后，将节点 A 的端口号和源 MAC 地址作为新的条目添加到 MAC 地址表中去。MAC 地址表一开始是空的，从空表的状态开始学习帧并不断添加新的数据。

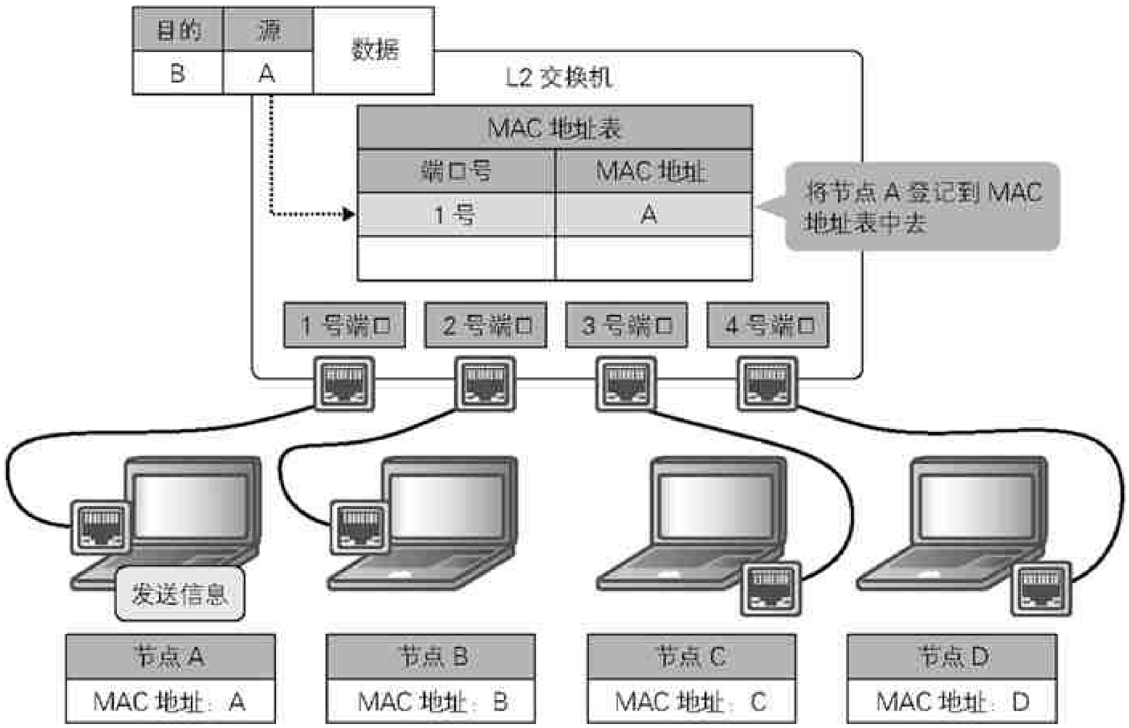


图 **2.1.12**　将节点 **A** 登记到 **MAC** 地址表中去

3 →　由于交换机并不知道节点 B 的 MAC 地址是多少，所以会将帧复制并发送给不与节点 A 相连接的所有端口。这种把帧同时发送给多个端口的现象叫作泛洪，是一种“我不知道是哪个 MAC 地址，所以干脆发给所有对象再说！”的应对方法。顺便提一句，广播的 MAC 地址 FF-FF-FF-FF-FF-FF 不会成为源 MAC 地址，因而不会被写入 MAC 地址表。也正因如此，广播经常会被泛洪。

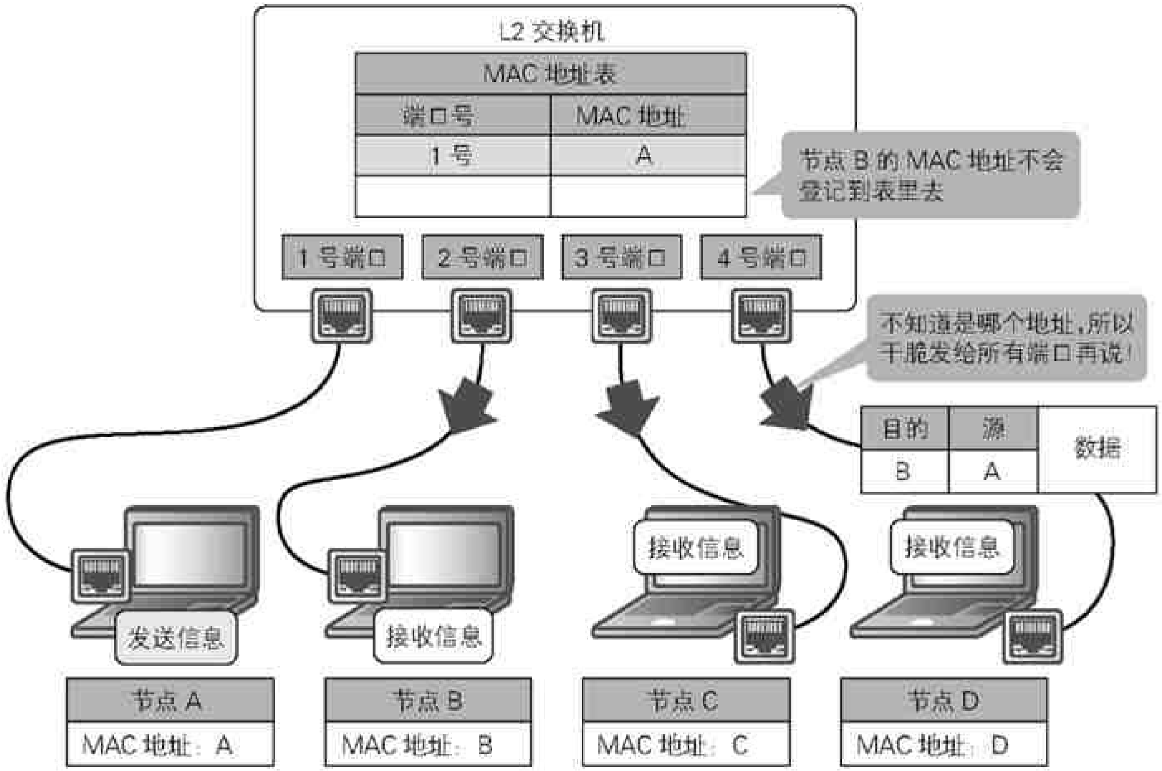


图 **2.1.13**　通过泛洪处理将信息发给所有端口

4 →　节点 B 收到帧之后认定这是发给自己的帧，为了回应节点 A，也会生成一个帧并传递给线缆。这时候，源 MAC 地址是节点 B 的 MAC 地址，目的 MAC 地址是节点 A 的 MAC 地址。节点 B 之外的节点 C 和节点 D 会判断出这是和自己无关的帧然后将其丢弃。

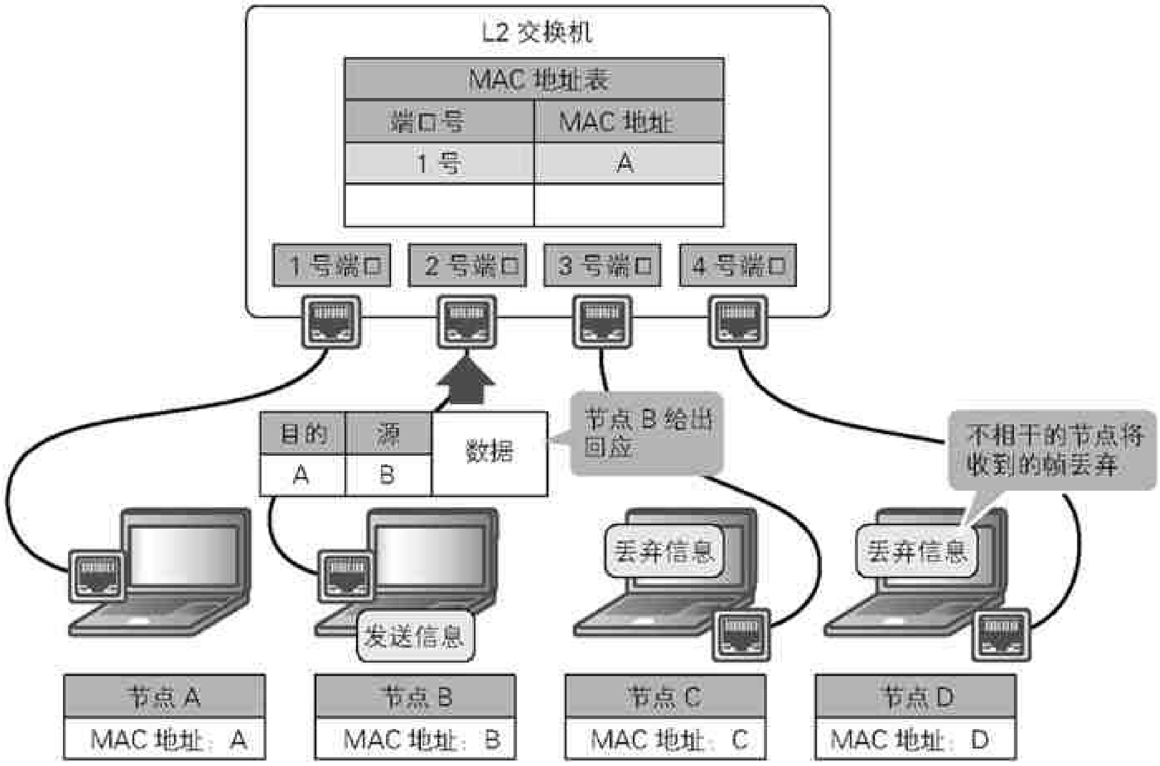


图 **2.1.14**　真正有关系的节点 **B** 会给出回应

5 →　交换机收到帧后，会将节点 B 的端口号和 MAC 地址登记到 MAC 地址表中。这样，节点 A 和节点 B 的 MAC 地址表就建成了。根据 MAC 地址表中的登记内容，发给节点 A 的帧会被马上转发到 1 号端口。MAC 地址表生成之后，节点 A 和节点 B 之间的帧转发就不会影响到其他节点，能够高效快速地进行了。

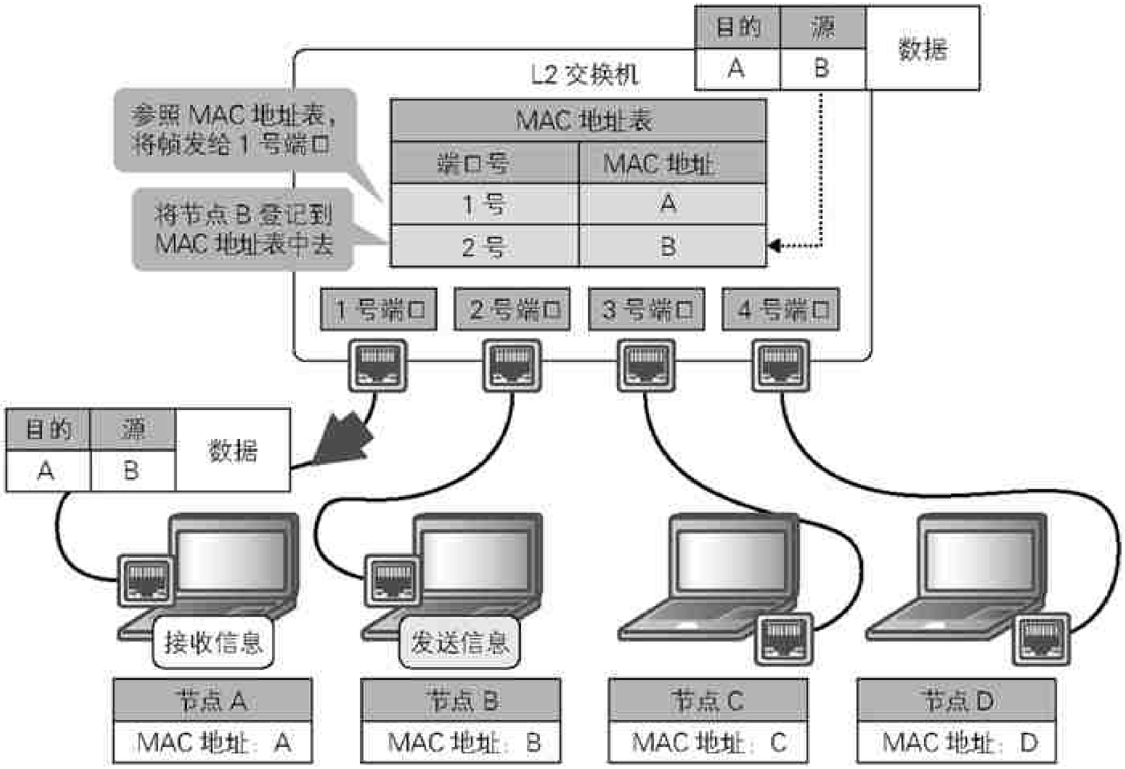


图 **2.1.15**　建立 **MAC** 地址表

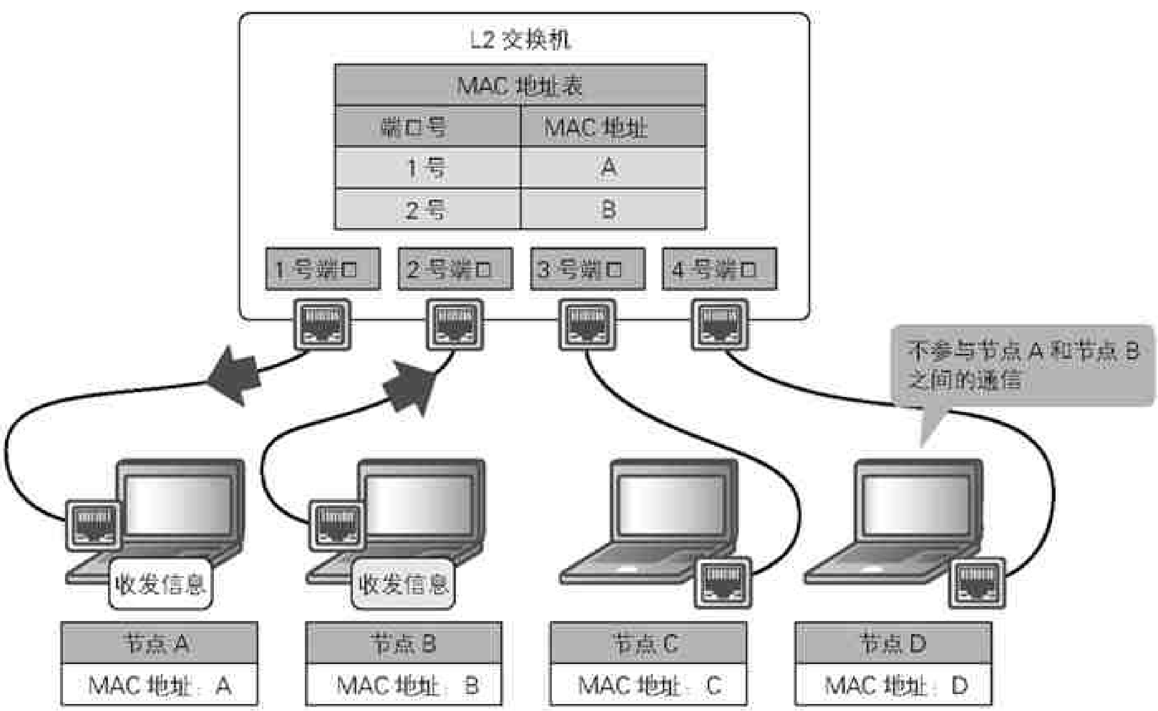


图 **2.1.16**　其他节点不会参与节点 **A** 和节点 **B** 之间的通信

6 → MAC 地址表建好之后，里面的条目并非是一直保留下去的。一直保留下去的话，内存再大也会不够用，而且节点一旦转移到别处去，与之对应的条目也就不起作用了。因此，与端口相连的线缆被拔掉时，条目会被删除；超过一定时间未收到帧时，条目也会被删除。从条目生成到条目被删除的时间叫作老化时间，思科 Catalyst 交换机的老化时间默认值为 300 秒（5 分钟），当然这个时间是可以修改的。

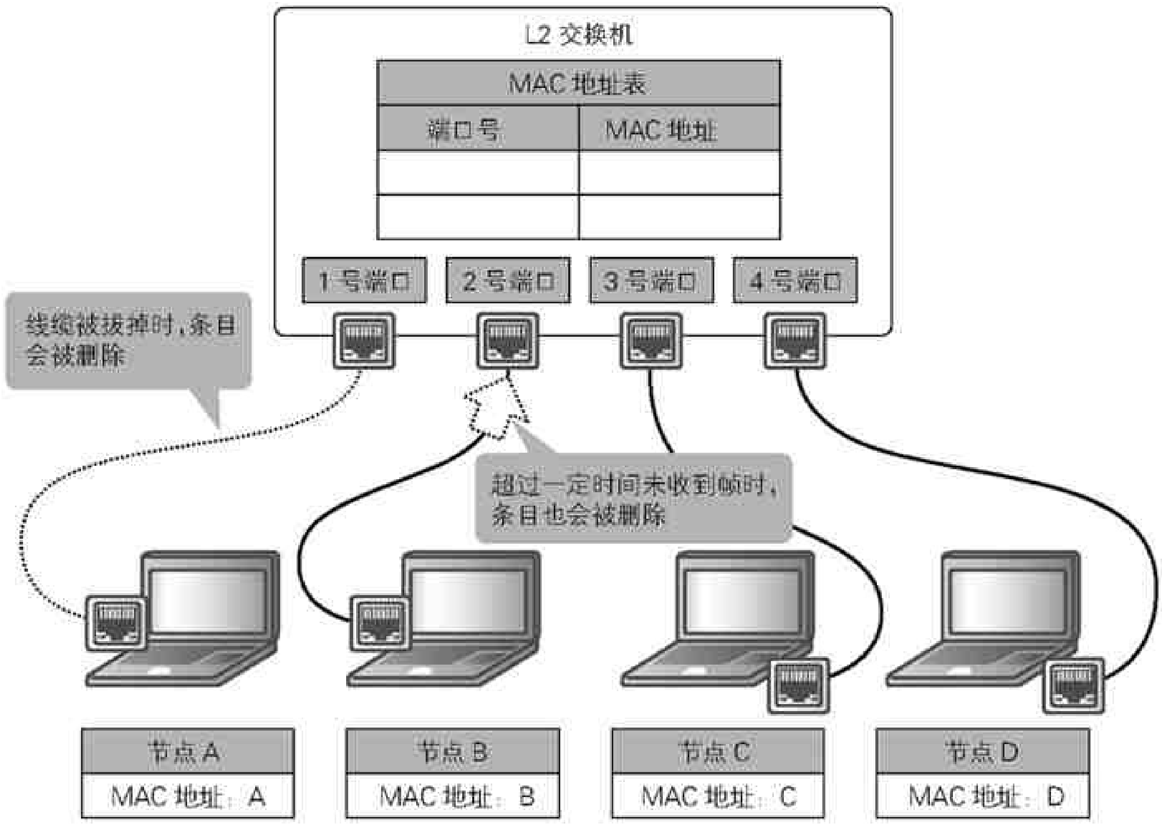


图 **2.1.17**　将不再需要的条目删除

查看 **MAC** 地址表

我们来实际看看下图中所示的 L2 交换机即 Switch1 的 MAC 地址表。

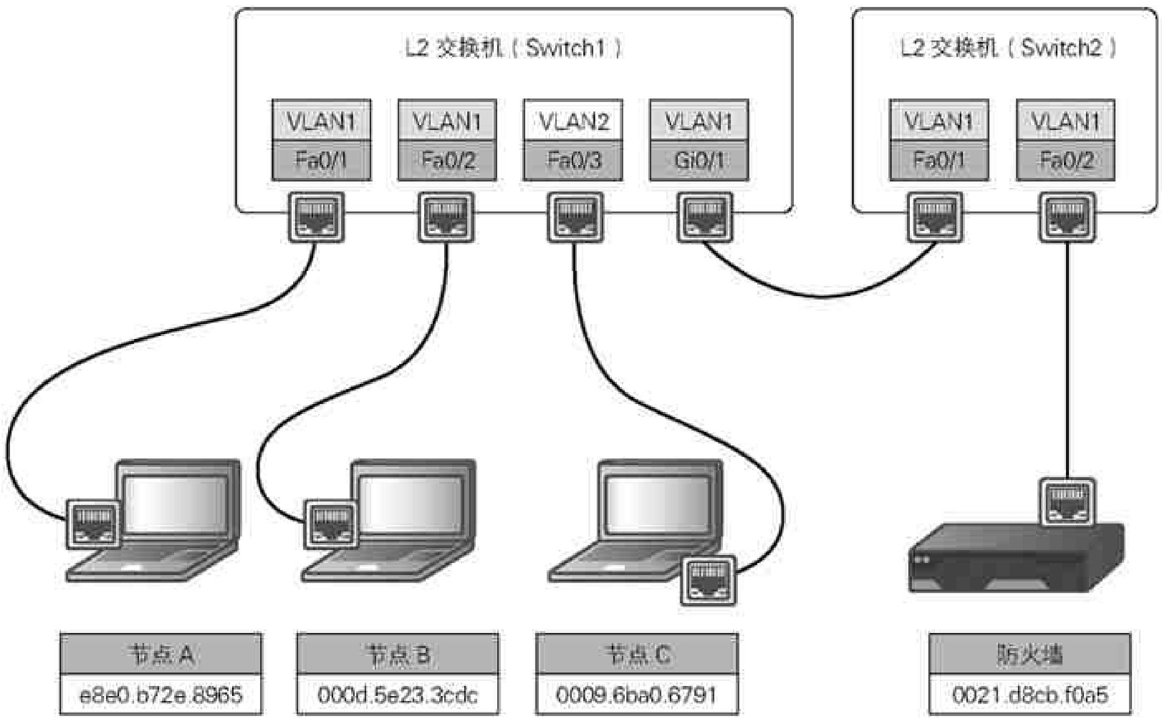
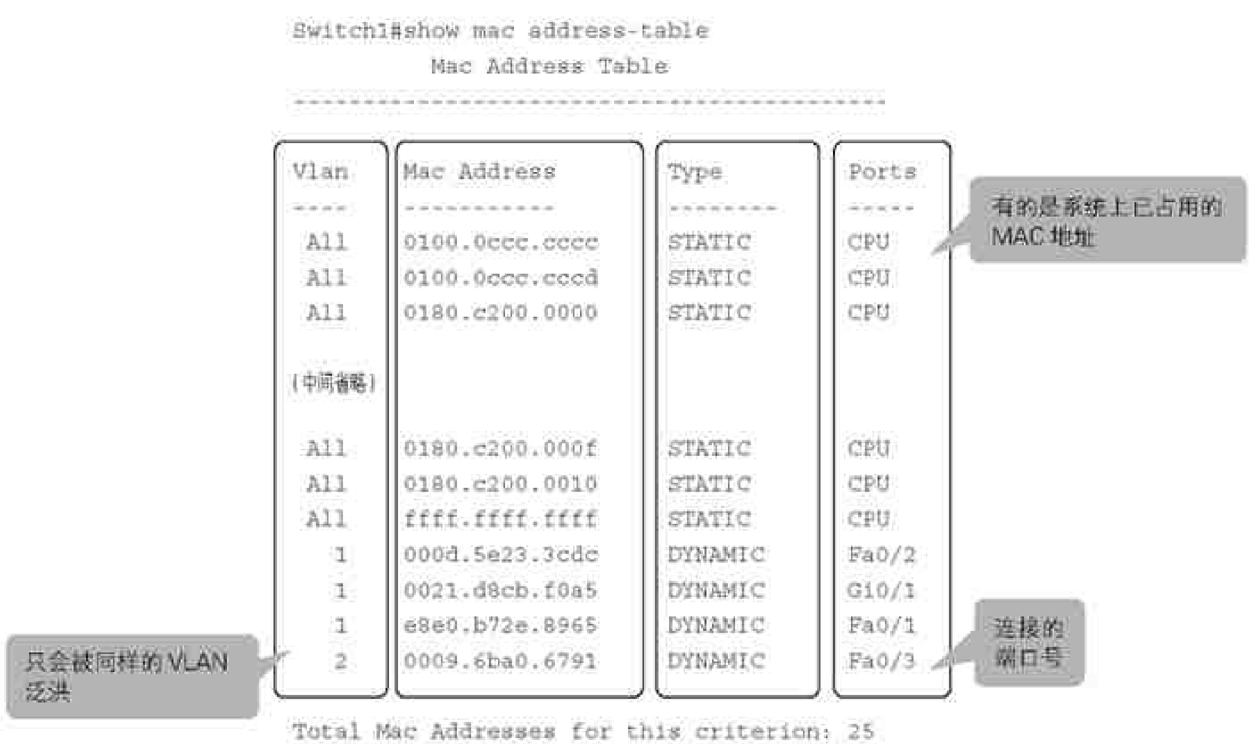


图 **2.1.18**　方便确认 **MAC** 地址表的连接图例

这里我们使用思科 Catalyst 交换机来确认。Catalyst 交换机可通过命令 show mac address-table 去查看 MAC 地址表。



```
Switch1#show mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
 All    0100.0ccc.cccc    STATIC      CPU
 All    0100.0ccc.cccd    STATIC      CPU
 All    0180.c200.0000    STATIC      CPU

（中间省略）

 All    0180.c200.000f    STATIC      CPU
 All    0180.c200.0010    STATIC      CPU
 All    ffff.ffff.ffff    STATIC      CPU
   1    000d.5e23.3cdc    DYNAMIC     Fa0/2
   1    0021.d8cb.f0a5    DYNAMIC     Gi0/1
   1    e8e0.b72e.8965    DYNAMIC     Fa0/1
   2    0009.6ba0.6791    DYNAMIC     Fa0/3
Total Mac Addresses for this criterion: 25
```

图 **2.1.19　Catalyst** 交换机中的 **MAC** 地址表示例

Mac Address 指节点的 MAC 地址，Ports 指连接的端口号。其他几个要素也在这里解释一下：Vlan 将在下一个小节里详细说明，广播和泛洪只会针对同一 VLAN 发送信息；Type 指 MAC 地址的类型，通过帧动态学到的内容为 DYNAMIC 型，超过老化时间条目就会被删除；STATIC 表示是本机静态设置的或是系统上已占用的 MAC 地址，不会被自动删除。

2.1.2.2　通过 **VLAN** 将广播域分隔开

广播能够波及的范围叫作广播域。用来搜索单播发送目的节点的 APR 是通过广播发送的，我们不妨认为广播域就是能够直接收发帧的范围。广播针对所有节点一气呵成地发送信息，似乎挺方便，然而帧会被发送到不相干的节点那里，从通信流量的角度来看效率并不高。这时候我们就要用到 VLAN（Virtual LAN，虚拟局域网），它是交换机所拥有的一项功能，能够将广播域分隔开，有利于提高通信效率。

有些人会把 VLAN 叫作网段、网络或 LAN（Local Area Network，局域网），称呼不同，但意思是完全一样的，我们不必太在意，知道他们指的是 VLAN 就可以了。

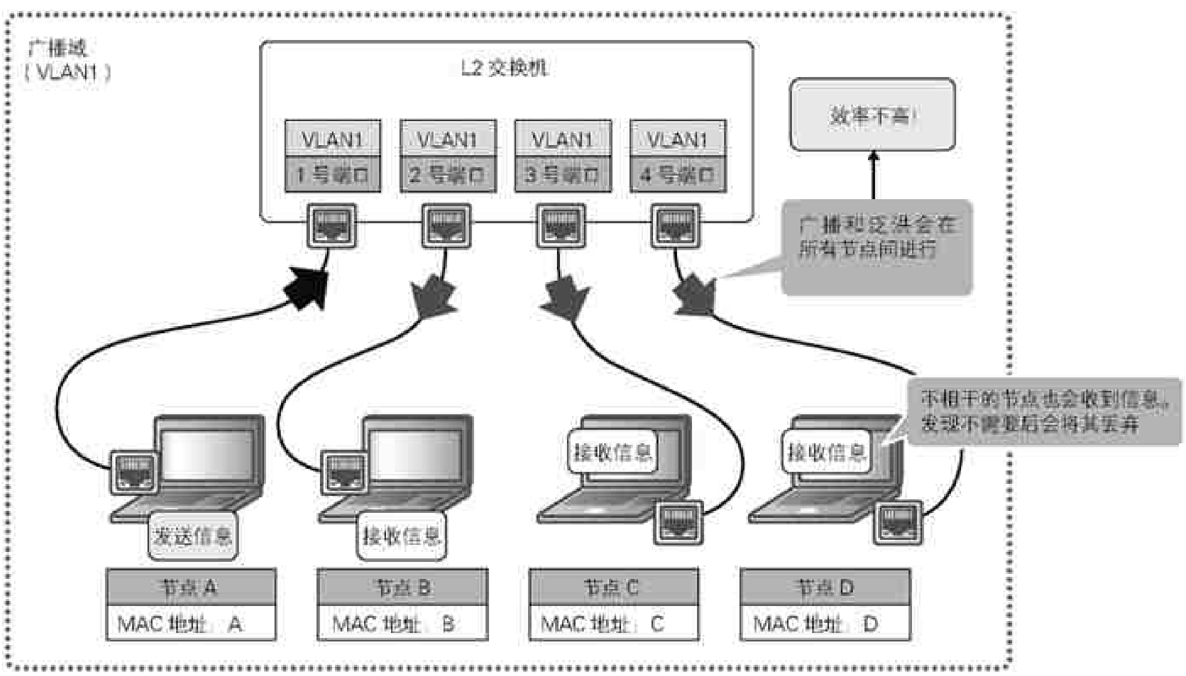


图 **2.1.20**　在同一个 **VLAN** 中，所有节点都会收到广播信息

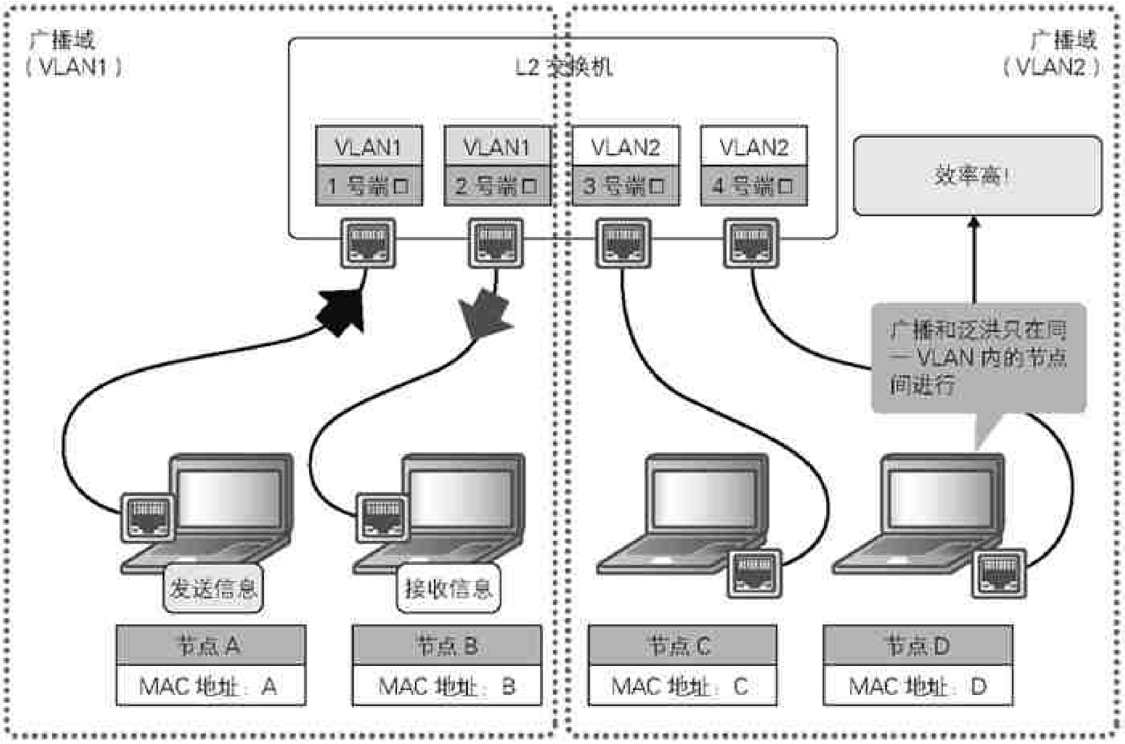


图 **2.1.21**　通过 **VLAN** 将广播域分隔开

VLAN 由数字构成

VLAN 能将广播域分隔开，这听起来似乎挺复杂，不过 VLAN 的本质其实就是代表 VLAN ID 的一些数字，其原理只是将 VLAN ID 的数字分配给各个端口，由此去识别端口而已。Catalyst 交换机最多可以支持 4096 个 VLAN ID，其中有几个已被系统保留，用于特殊的目的。机型和 OS 版本不同，实际上能够使用的 VLAN ID 范围和个数也就不同，这点大家一定要仔细确认。

表 **2.1.3**　有些 **VLAN ID** 已被系统占用，不能使用

VLAN ID	用途
0	系统已占用的VLAN

1	默认VLAN
2～1001	用于以太网的VLAN
1002～1005	FDDI，令牌环网的默认VLAN
1006～1024	系统已占用的VLAN
1025～4094	用于以太网的VLAN
4095	系统已占用的VLAN

那么，我们应该如何设置 VLAN 呢？这个也并不复杂。VLAN 只有两种设置方法，不是端口 VLAN 就是打标 VLAN。下面就详细说明这两种设置方法。

端口 **VLAN** 设置是让端口和 **VLAN** 一一对应

正如其名，端口 VLAN 是将 VLAN 分配给端口的设置方法，有静态设置（静态 VLAN）和动态设置（动态 VLAN，根据基本规则的要求动态地进行设置）之分，以使用前者的环境居多。动态 VLAN 一般仅用于 CCNA、CCNP 等思科公司的测试中，本书只讲解静态 VLAN。

静态 VLAN 会将 VLAN ID 分配给端口。将 VLAN ID 分配给各个端口，就完成了对一台交 机的逻辑划分，也就完成了对广播域的划分。举个例子，将 VLAN1 和 VLAN2 分配给端口之后，VLAN1 所属终端的广播域和 VLAN2 所属终端的广播域就不一样了，VLAN2 不会收到 VLAN1 发出的广播，也不会收到 ARP，二者之间无法进行直接通信。如果要让 VLAN1 和 VLAN2 之间产生通信，必须通过 L3 交换机或路由器等 L3 设备进行中转才行。设置端口 VLAN 时也可以跨交换机进行，但必须注意一点，那就是有多少 VLAN 就得准备多少端口和线缆去连接交换机。

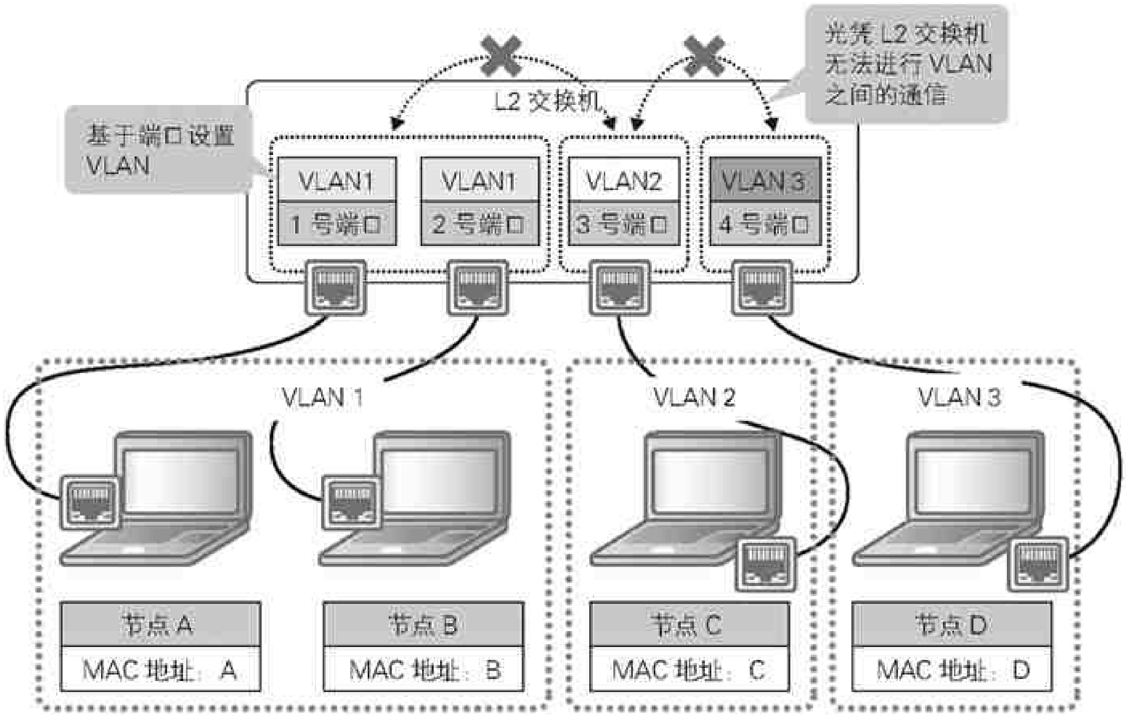


图 **2.1.22**　基于端口设置 **VLAN**

打标 **VLAN**

正如其名，打标 VLAN 是给 VLAN 打上标签的设置方法。打上标签？乍一听你也许会觉得丈二和尚摸不着头脑，但实际上它只是给帧打上一个包含 VLAN 信息的标签而已。有两种打标方法，一种是能够传输非以太网帧的 ISL 方式，另一种是只能传输以太网帧的 IEEE802.1Q 方式。现在的网络环境大多是以太网，因此打标 VLAN 大多采用 IEEE802.1Q 方式，IEEE802.1Q 也是 IEEE 委员会制定的打标 VLAN 的标准规格，而 ISL 我们只能在 CCNA、CCNP 等思科公司的测试中见到。本书只讲解 IEEE802.1Q。

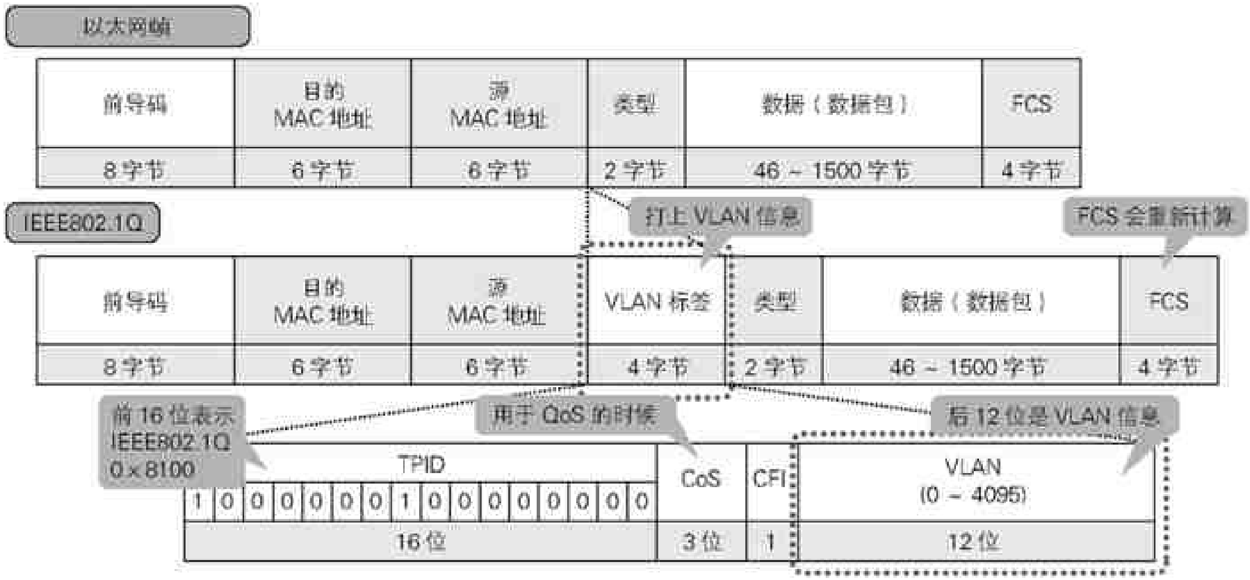


图 **2.1.23**　通过 **IEEE802.1Q** 为 **VLAN** 打标

端口 VLAN 必须是一个端口对一个 VLAN，所以如果要跨交换机进行设置以促成同一 VLAN 内节点之间的通信，那么有多少 VLAN 就要准备多少端口和线缆。但是，这种情况在可扩展性上会发生问题，假设有 1000 个 VLAN，那么我们就得准备 1000 根线缆和 1000 个端口，无论是布线还是设置都极其耗时费力，端口再多也会有不够的时候。

这时候我们就要用到打标 VLAN。交换机端口发送信息时给帧打上一个含有 VLAN 信息（VLAN ID）的 VLAN 标签，目的端口收到帧时先卸掉标签，然后再交给需要该信息的节点。通过打标 VLAN 能够识别收到了哪个 VLAN 的帧，所以只需一根线缆和一个端口即可，布线也十分简单。使用打标 VLAN 时，相连的两台设备必须能够识别出同样的信息才行，因此两边设置的 VLAN ID 务必要统一。

最近，打标 VLAN 不止用于交换机之间的连接，也越来越多地用在虚拟环境（VMware 环境）中。在虚拟环境中，隶属于各种 VLAN 的虚拟机共存于一个物理服务器中，通过虚拟交换机出现在网络上。连接 L2 交换机时，针对需要连接的端口，我们必须要设置该虚拟机所属的所有 VLAN，这时候就要用到 IEEE802.1Q 了。

在虚拟环境中，虚拟机隶属于一个叫作端口组的虚拟交换机端口设置组，通过 vSwitch（虚拟交换机）连到网络上。假设虚拟机分别隶属于叫作 VLAN10 和 VLAN20 的端口组，那么映射到 vSwitch 上 的

vmnic（物理网卡）就隶属于 VLAN10 和 VLAN20。vSwitch 给 VLAN10 和 VLAN20 打上标签，将与 vmnic 连接的交换机端口划分到打标 VLAN。用 vSwitch 进行的标签处理叫作 VST（Virtual Switch Tagging，虚拟交换机标记）。

另外，对思科 Catalyst 交换机而言，打标 VLAN 一般会称为 trunk，而对其他生产商生产的机器而言，trunk 则指的却是链路聚合功能以及由该功能衍生出来的逻辑端口。二者很容易混淆，请大家务必区分清楚。关于链路聚合的内容将在 4.1.1.1 节中详细说明。

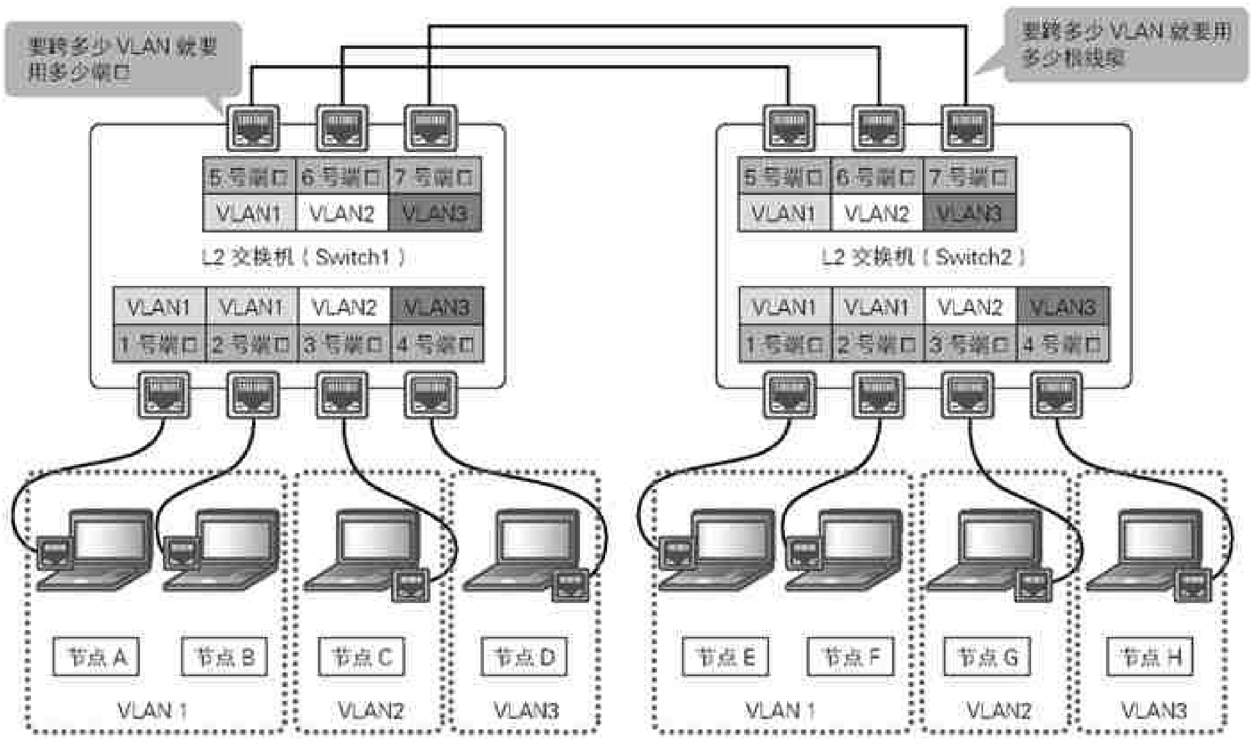


图 **2.1.24**　使用端口 **VLAN** 时，如果 **VLAN** 要跨交换机会非常麻烦

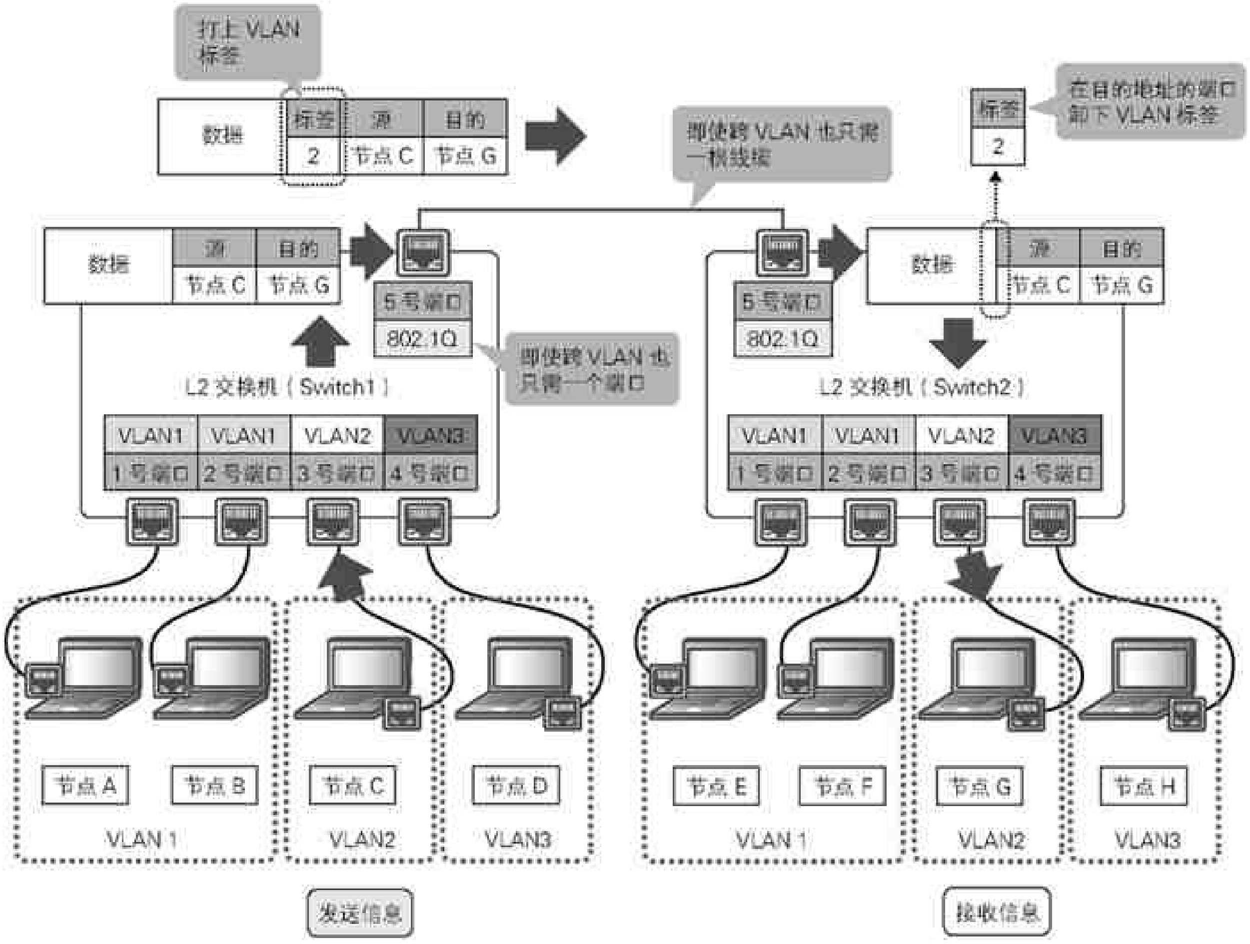


图 **2.1.25**　使用打标 **VLAN** 时，只需一根线缆和一个端口即可

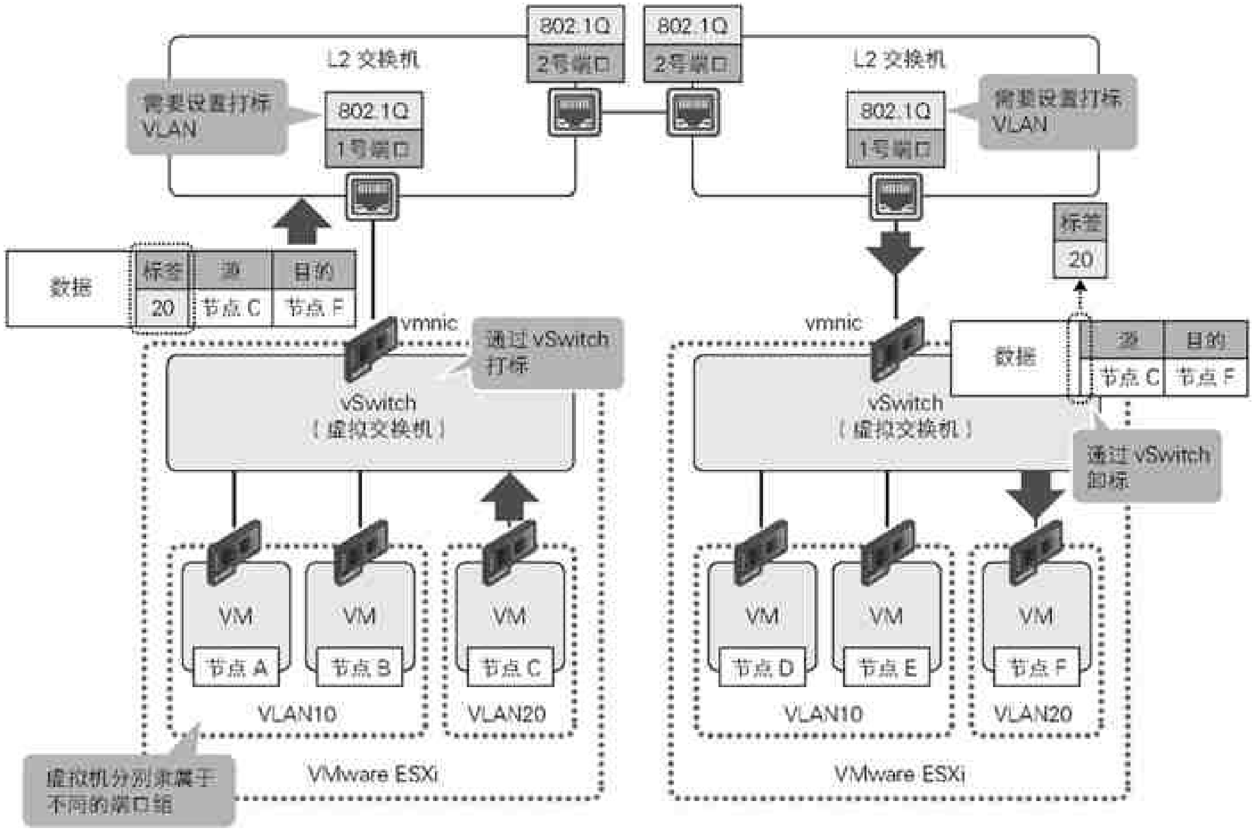


图 **2.1.26**　虚拟环境中也经常会用到打标 **VLAN**

※ 实际上 L2 交换机中也有对 VLAN 标签的处理，图中省略未画。

使用打标 **VLAN** 时应统一本征 **VLAN**

我们在设置打标 VLAN 时要注意本征 VLAN 的存在。设有 IEEE802.1Q 的端口并非对所有的 VLAN 都会打标，有一个 VLAN 是除外的，那就是本征 VLAN。

这里的关键在于要保持相邻设备的本征 VLAN 一致。也许你会想，如果是不打标的 VLAN 就可以不用在意嘛，然而实际情况却并非如此。假设一边的本征 VLAN 是 VLAN1 而另一边的本征 VLAN 是 VLAN3，那么它们会分别映射不同的广播域，二者之间就无法进行通信了。所以我们不仅要保持两边的 VLAN ID 一致，还要保持两边的本征 VLAN 也一致才行。

Catalyst 交换机的默认本征 VLAN 是 VLAN1，当然我们也可以修改这一默认设置。另外，只有在使用 CDP（Cisco Discovery Protocol，思科

发现协议）的前提下，才能自动检测出两边设备的本征 VLAN 不一致并发出警告信息“%CDP-4-NATIVE_VLAN_MISMATCH”。关于 CDP 的内容将在 5.1.4.1 节中说明。

本征 VLAN 在虚拟环境中的原理也一样。如果我们在端口组的 VLAN ID 中选择“无（0）”，那么隶属于该端口组的虚拟机的帧就不会被打标。不被打标也就等同于本征 VLAN。而如果在这里设置 VLAN ID，那么 vSwitch 就会给该 VLAN ID 打标。

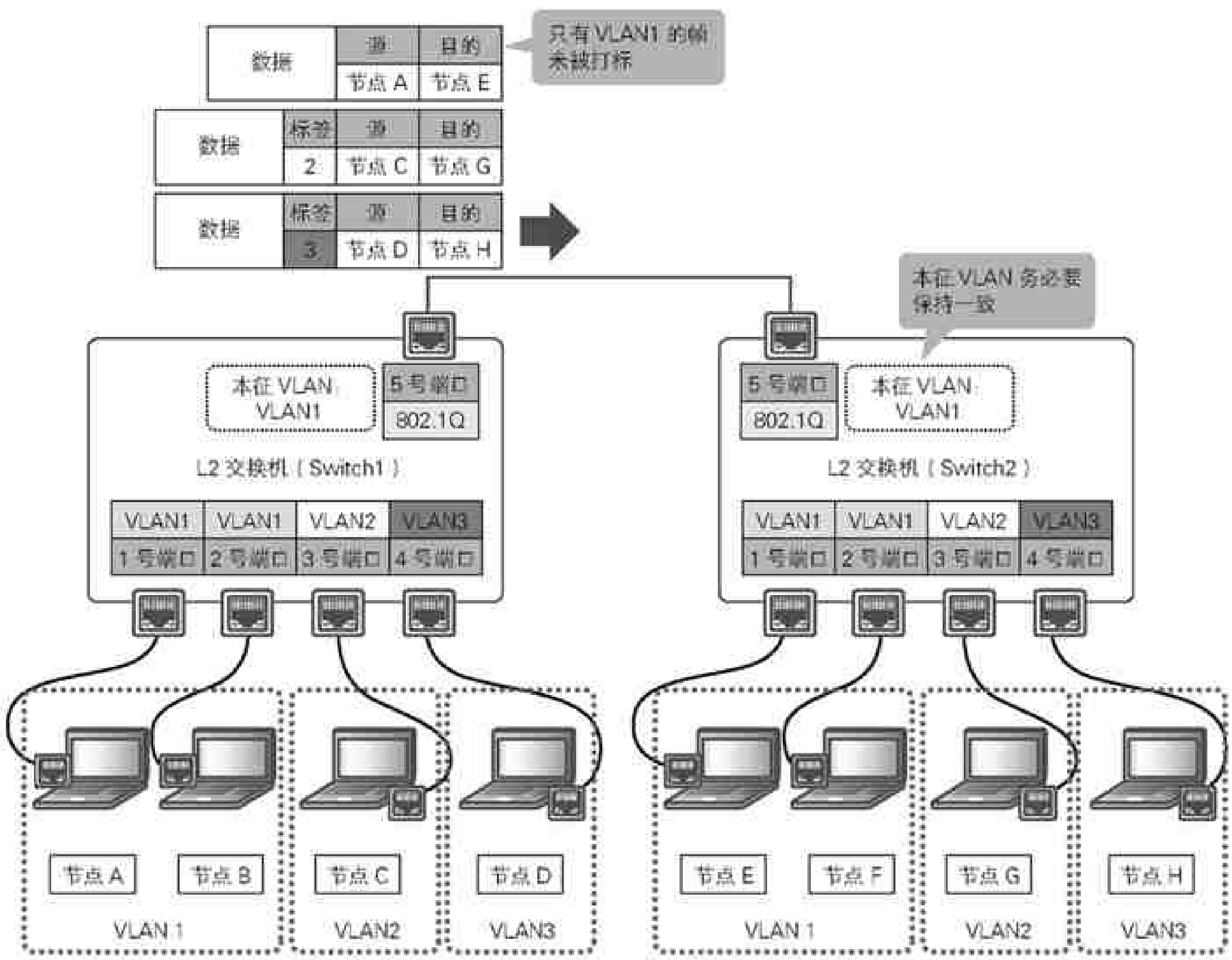


图 **2.1.27**　有一个 **VLAN** 不会被打标

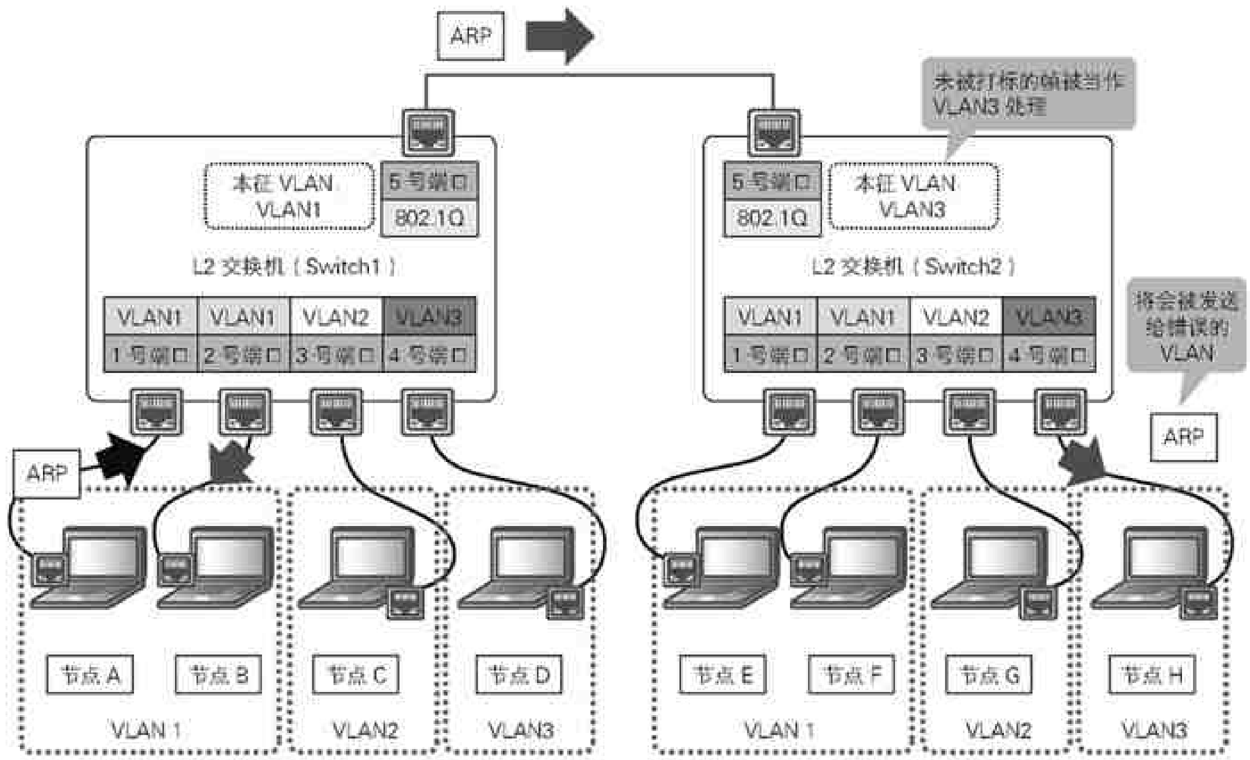


图 **2.1.28**　两边的本征 **VLAN** 如果不一致就无法正常通信（以 **ARP** 为例）

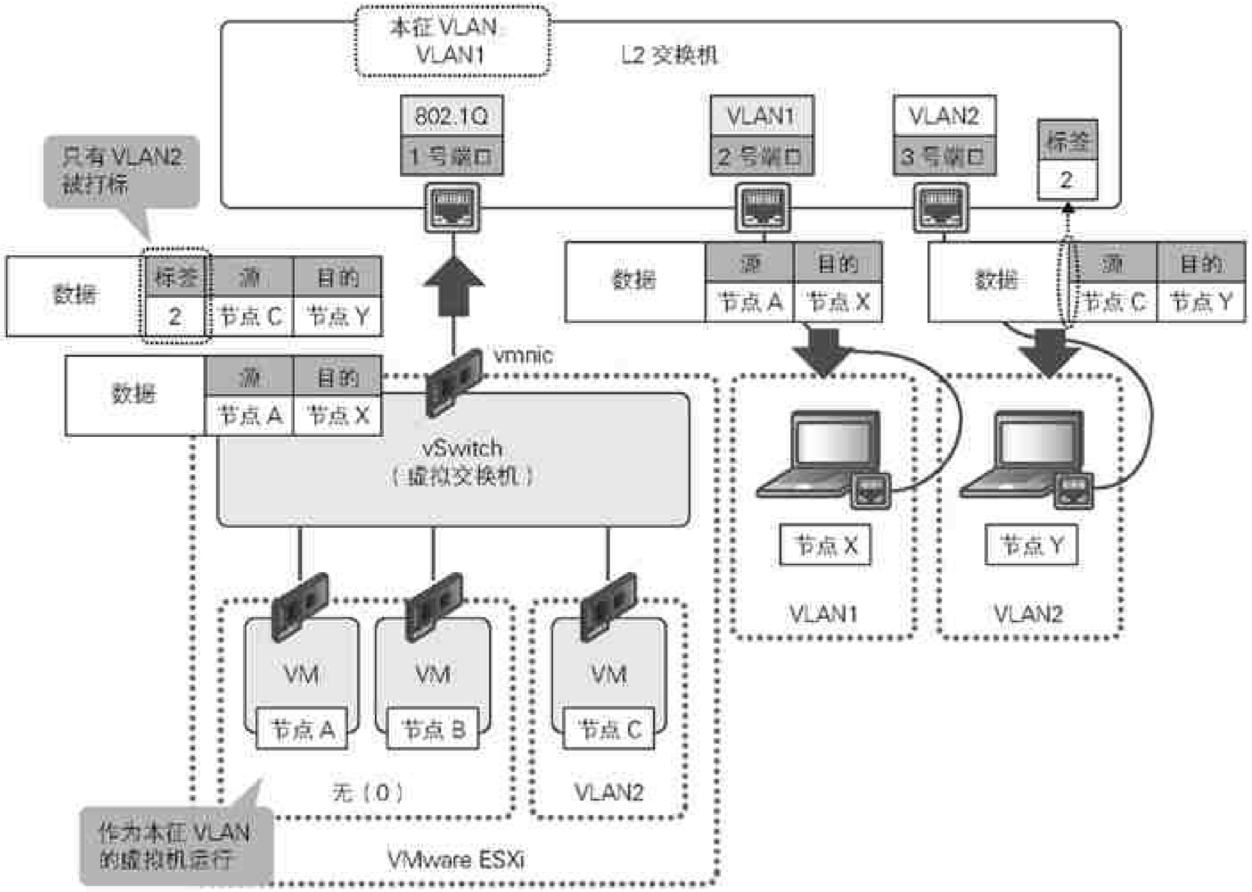


图 **2.1.29**　在虚拟环境中也要注意本征 **VLAN**

2.1.3　ARP 将逻辑和物理关联到一起

在网络的世界里只有两个概念是表示地址的，一个是前面一直在讲的 MAC 地址，另一个则是 IP 地址。MAC 地址是硬件被赋予的物理地址，在数据链路层（L2）中发挥作用。IP 地址则是由 OS 设置的逻辑地址，在网络层（L3）中发挥作用。这两个地址如果步调不一致就会乱套，务必让它们彼此协调配合才行。ARP（Address Resolution Protocol，地址解析协议）就是能让这两个地址保持协调的存在，它在物理和逻辑之间起着桥梁的作用。

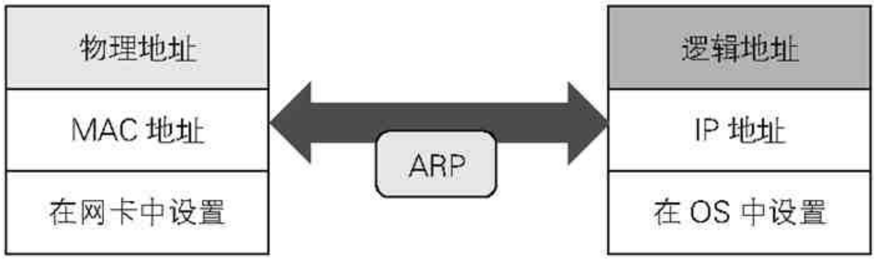


图 **2.1.30　ARP** 是物理和逻辑之间的桥梁

2.1.3.1　ARP 通过 **IP** 地址查询 **MAC** 地址

ARP 是物理和逻辑之间的桥梁，这听起来似乎非常高深，不过它实际上只是将 IP 地址和 MAC 地址关联起来而已，并没有进行什么复杂的处理。

我们知道，收到来自网络层的 IP 数据包之后，节点必须将其封装成帧并传递给线缆。然而，刚刚收到 IP 数据包时节点并不清楚该如何对它进行封装，因为节点虽然知道源 MAC 地址就是本机的 MAC 地址，却不知道目的 MAC 地址是什么。这时候就要用到 ARP 了。ARP 先去查看 IP 数据包的目的 IP 地址，如果是同一网段的节点，ARP 就去查询该 IP 地址的 MAC 地址；如果是不同网段的节点，ARP 就去查询默认网关的 MAC 地址。默认网关相当于一个通往非本地网段的出口，如果数据包是发给非本地的其他网段而且并不清楚目的 IP 地址，那么它就会被发给默认网关。这里说的“网段”指的是 IP 地址的所属范围，会在 2.1.1.2 节中另外详细说明。在目前这个阶段，我们可以认为广播域和 VLAN 是同一个意思。

这里请记住一点：ARP 是通过 IP 地址去查询 MAC 地址的。

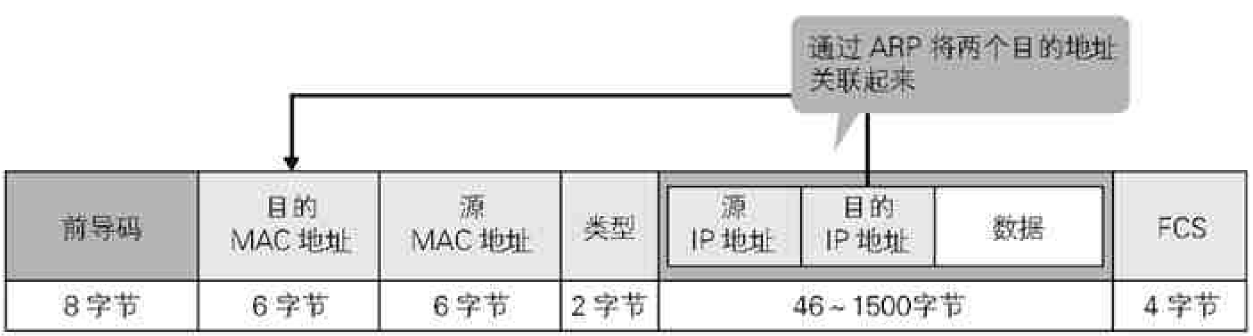


图 **2.1.31　ARP** 将两个地址关联到一起

ARP 的原理很简单

ARP 的原理很简单，非常容易理解。请想象一下这个场景：你大声地（广播）问大家“某某某是哪位啊？”，于是某某某回答“我就是某某某啊！”。

这个“某某某是哪位啊？”的提问叫作 ARP Request（ARP 请求）。“某某某”的部分就是从网络层发来的数据包中目的地的 IP 地址。ARP 请求通过广播散发出去，同一 VLAN 中的所有节点都会收到该请求。

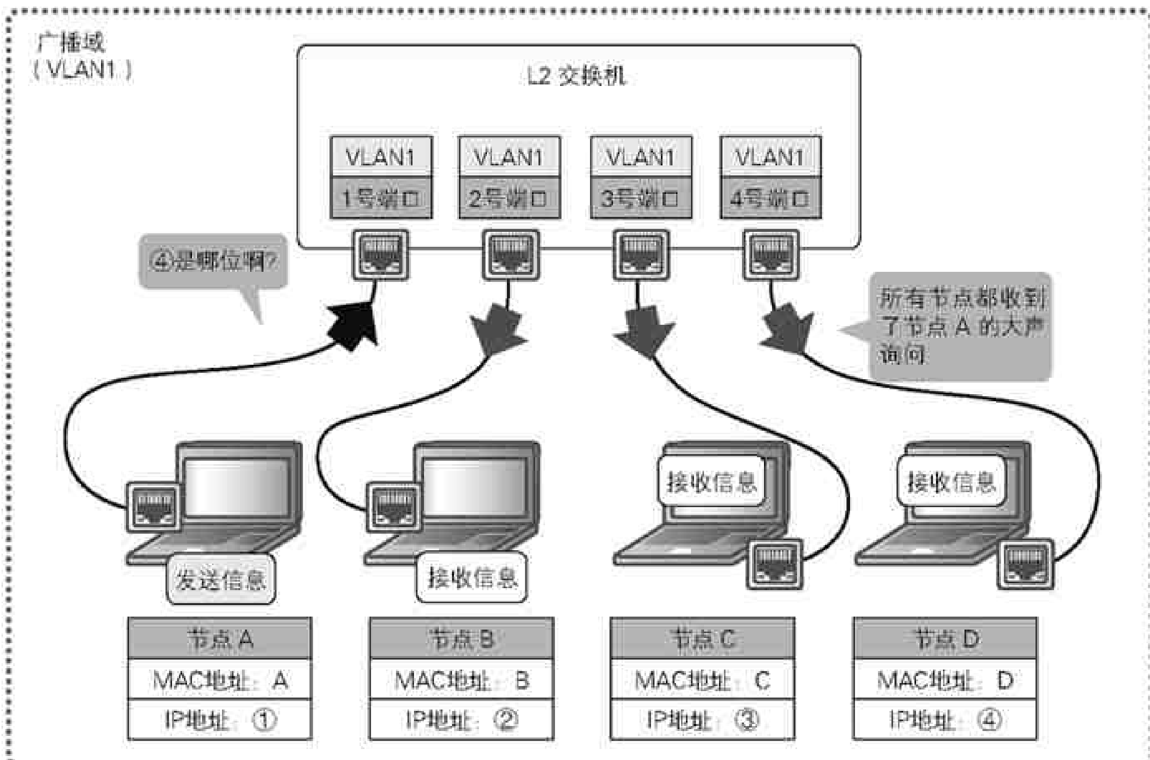


图 **2.1.32**　**ARP** 不知道应将信息发往何处，于是大声询问所有节点

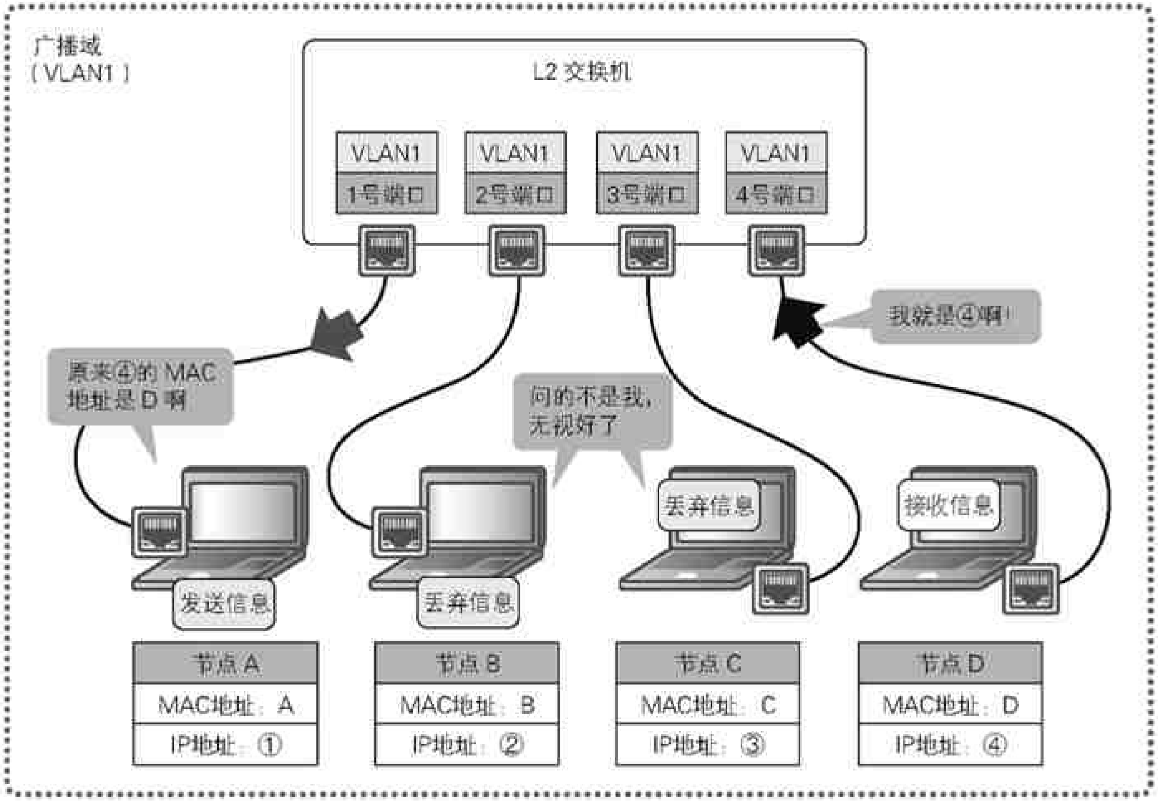


图 **2.1.33**　只有真正相关的节点给出回应

针对 ARP 请求返回的这个“我就是某某某啊！”的回应叫作 ARP Reply（ARP 回复）。在同一 VLAN（广播域）的所有节点中，只有一个真正拥有对象 IP 地址的节点会做出回应。其他的节点因为与己无关，会将收到的信息丢弃。ARP 回复不需要使用广播，它通过单播将信息发给发送 ARP 请求包的节点。

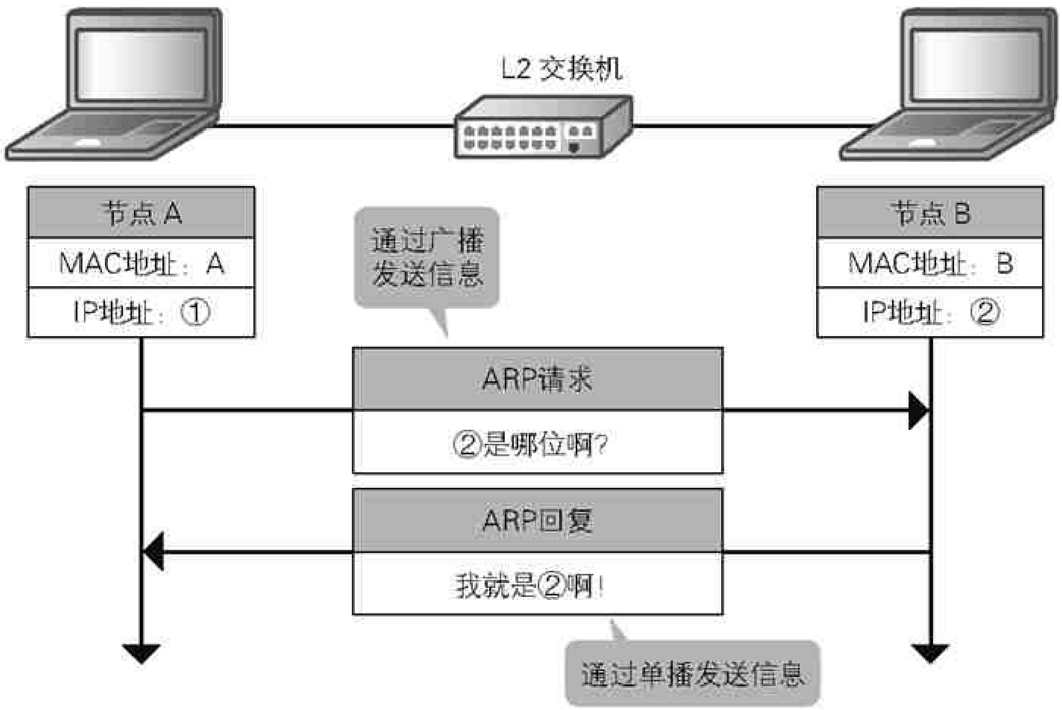


图 **2.1.34**　通过广播和单播查出对方节点的 **MAC** 地址

以太网的数据部分中包含着诸多的地址信息

ARP 在以太网的数据部分写入了很多地址信息，并以此将 MAC 地址和 IP 地址关联起来。由于操作（Operation）和地址之外的数据都是固定不变的，因此本书只介绍操作和地址这两项内容。

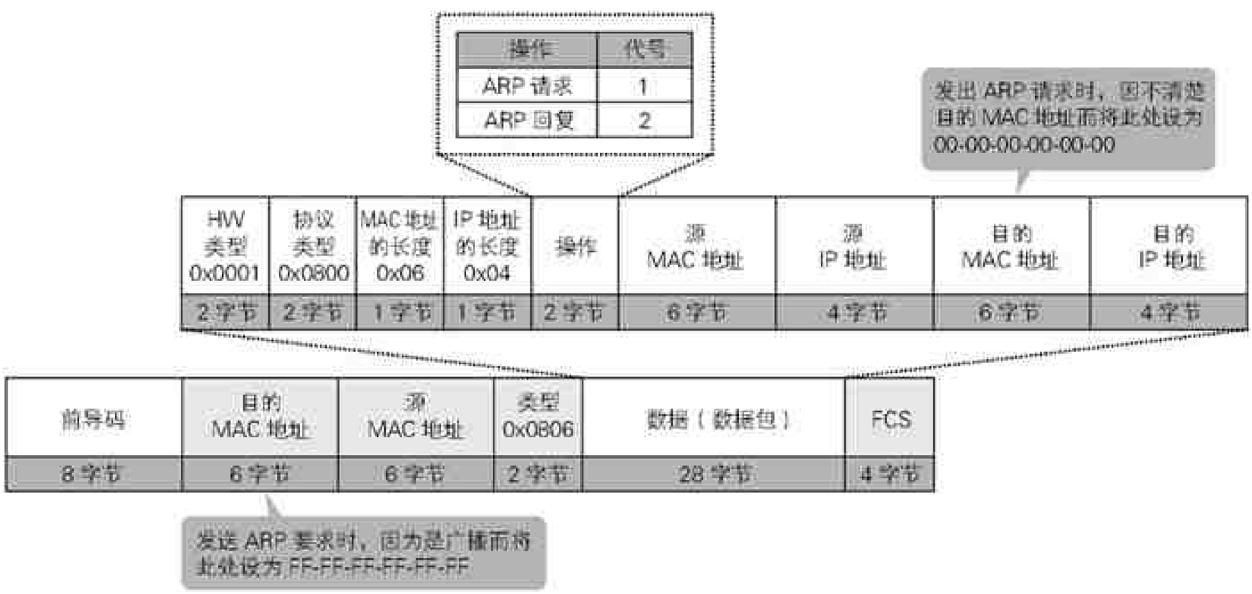


图 **2.1.35**　**ARP** 在数据部分写入了很多信息

ARP 请求的操作代号是 1。请求时，节点将源 MAC 地址和源 IP 地址设置为本机地址，将目的 MAC 地址设置为 00-00-00-00-00-00（因为尚不清楚实际地址是什么），将目的 IP 地址设为欲查询其 MAC 地址的节点的 IP 地址。ARP 请求采用广播的形式，收到广播的所有节点都会去查看目的 IP 地址，但是只有真正拥有该 IP 地址的节点才会给出 ARP 回复，拥有其他 IP 地址的节点则会将收到的信息丢弃。

ARP 回复的操作代号是 2。回复时，节点将源 MAC 地址和源 IP 地址设置为本机地址，将目的 MAC 地址和目的 IP 地址分别设置为发送了 ARP 请求的节点的 MAC 地址和 IP 地址。ARP 回复采用单播的形式，向发来 ARP 请求的节点返回信息。收到 ARP 回复的节点查看源 MAC 地址和源 IP 地址的部分后，就能知道对方的 MAC 地址是多少，而知道 MAC 地址之后就可以开始通信了。将 MAC 地址和 IP 地址关联起来的表叫作 ARP 表。

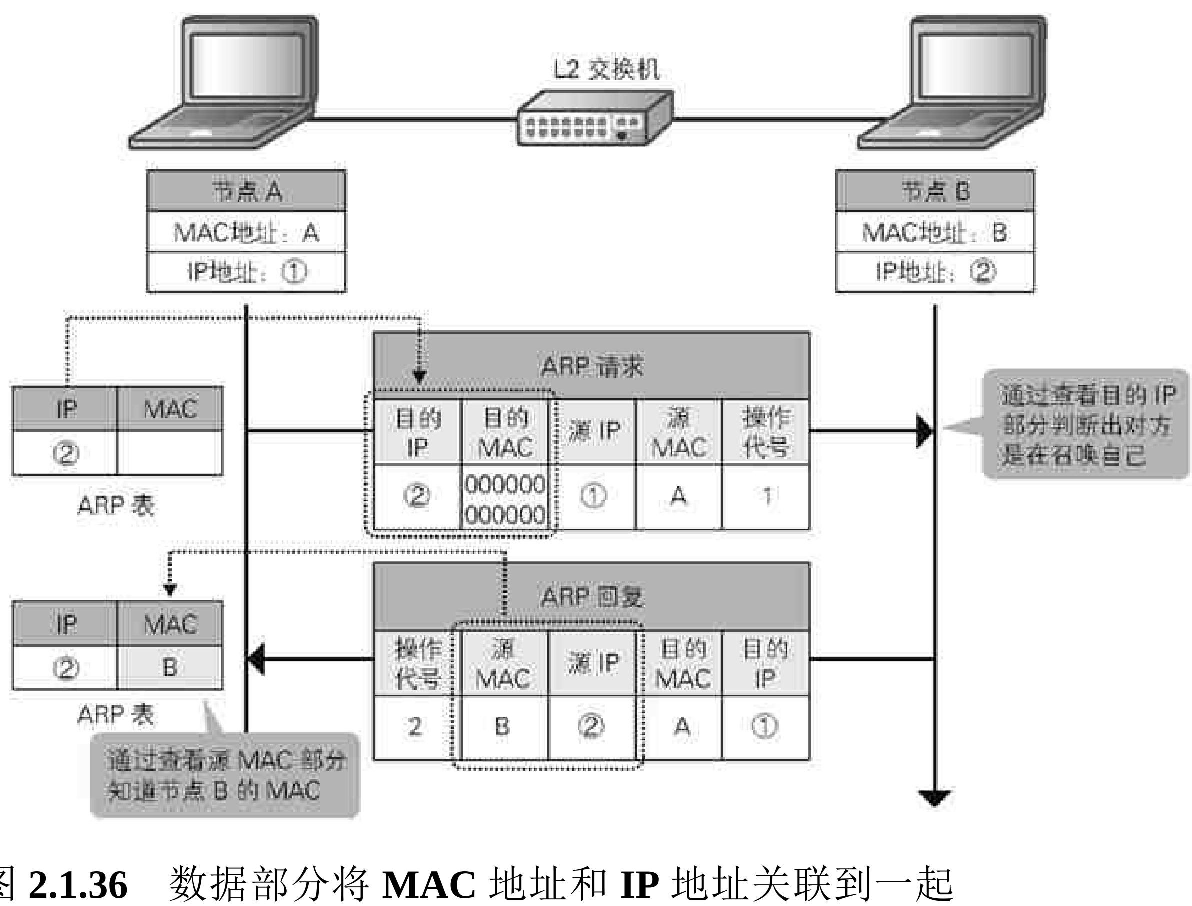


图 **2.1.36**　数据部分将 **MAC** 地址和 **IP** 地址关联到一起

通过高速缓存控制通信流量

读到这里，想必各位已经能够了解 ARP 在通信中的重要作用了。对于通信来说，ARP 是基础中的基础。正是因为通过 ARP 查询出发送目的地的 MAC 地址，通信才得以成立。

然而 ARP 有着致命的弱点，那就是“以广播发送为前提条件”。由于一开始发送方并不知道对方节点的 MAC 地址，使用广播也算是一种必然的选择。然而广播会向同一网段中的所有节点都发送数据，是一种效率很低的通信。假设 VLAN 中有 1000 台节点机，那么通信就会流向这 1000 台机器，假如每台节点机通信时都发送 ARP，那么仅 ARP 通信流量就会充斥于整个 VLAN。原本 MAC 地址和 IP 地址就不会频繁发生变化，于是人们为 ARP 开发出了可暂时存储条目的高速缓存功能。

ARP 查询到对方节点的 MAC 地址后，会在 ARP 表中添加新的条目并将其暂时保存。在暂时保存期内 ARP 不会发送信息，超过一定时间（时限）之后则会删掉该条目并发送 ARP 请求。时限因 OS 而异，Windows 7 是 10 分钟，思科公司的设备则为 4 个小时，当然，这些默认时限都是可以修改的。

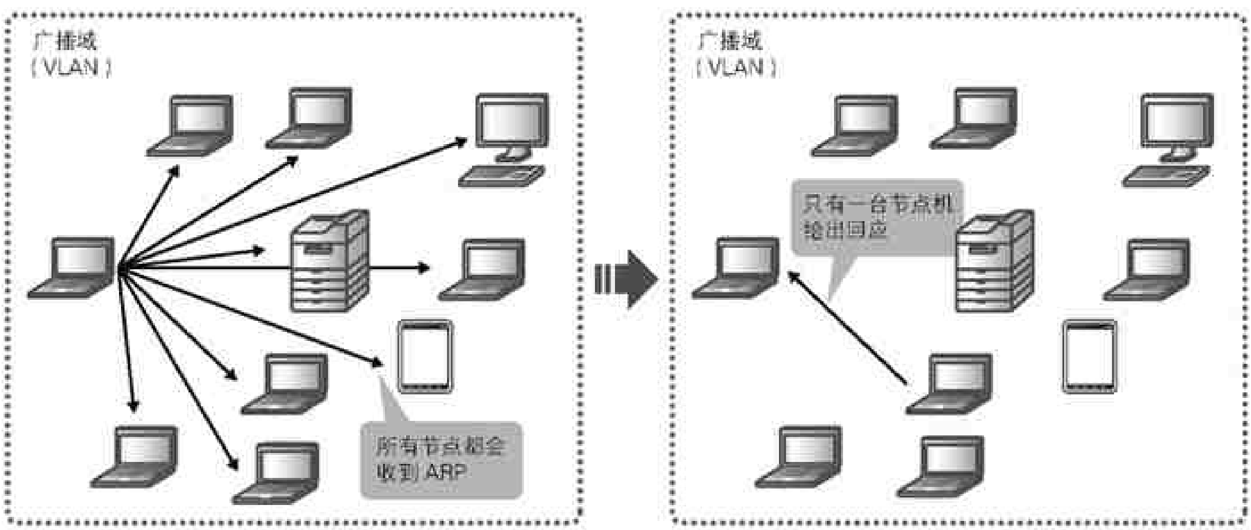


图 **2.1.37　ARP** 将信息发给所有节点，但只能收到其中一台节点机的回应，效率极低

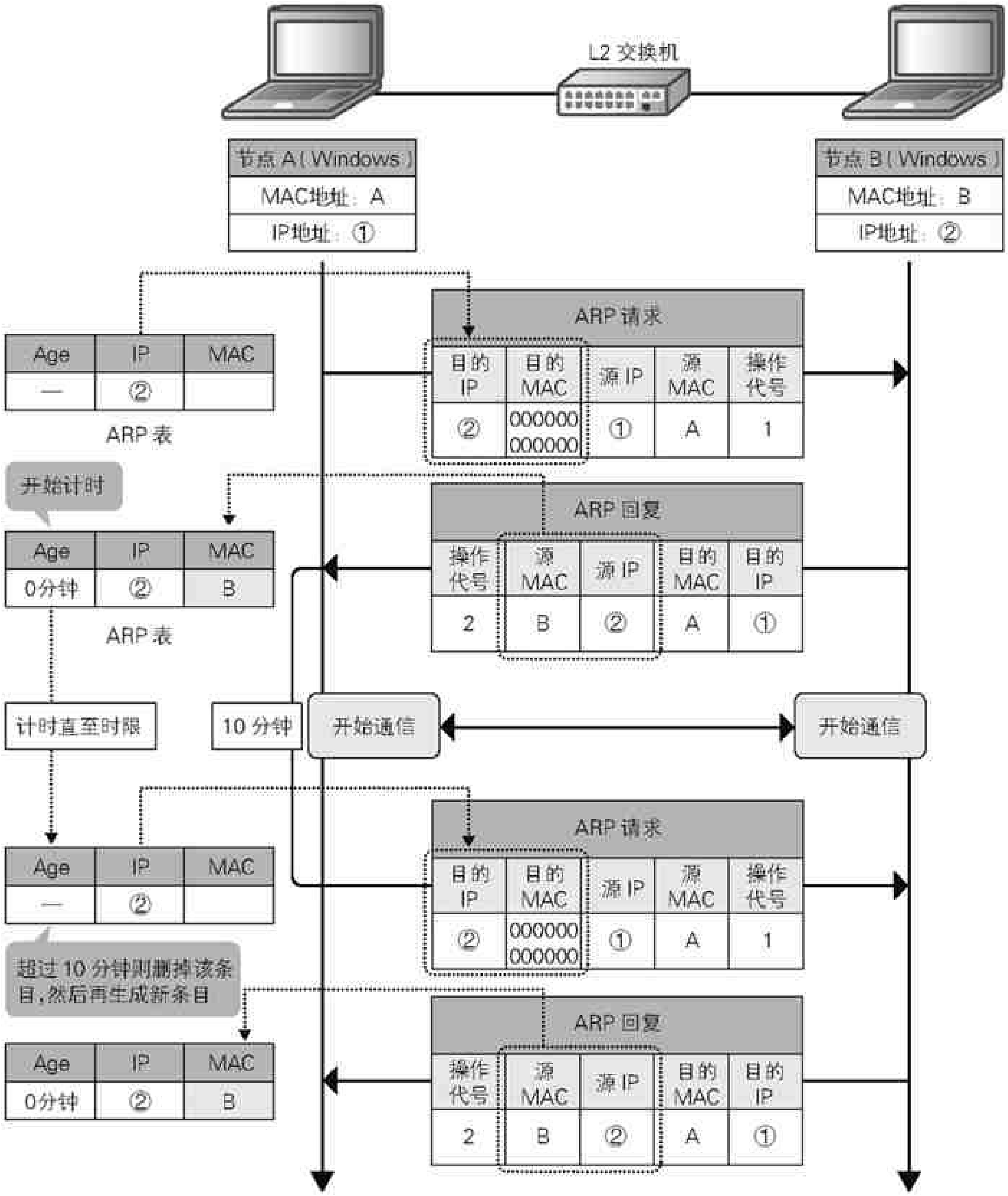


图 **2.1.38**　通过高速缓存功能提高效率

更换设备时要注意 **ARP** 高速缓存

高速缓存功能在减少 ARP 的通信流量上发挥着巨大的作用，但它并非是完美无缺的。有了这项功能，实时性就会差强人意，这是拥有高速缓存功能的所有协议相通的一个致命弱点。当 IP 地址或 MAC 地址发生变化时，ARP 表依然保留着旧的信息，因此只有等保留的条目超过时限后，ARP 再重新学习一遍新地址才能重新开始通信。

那么，节点的 MAC 地址或 IP 地址会频繁发生变化吗？不会。在大多数情况下 MAC 地址和 IP 地址都是一直不变的，因此，一般情况下我们不必在意 ARP 的这个弱点，需要注意的是更换设备的时候。例如，网络打印机突然出故障而不得不换掉了，这时候 IP 地址还是不变的，但 MAC 地址会变，然而打印机周边的节点并不知道 MAC 地址的这个变化，还是会去查看 ARP 表，试图和已经不存在的 MAC 地址通信，直到表中该条目的时限到来为止（超过时限之后才能重新开始通信）。要想解决这个问题就得使用 GARP，关于 GARP 的内容将在 2.1.3.3 节中详细说明。

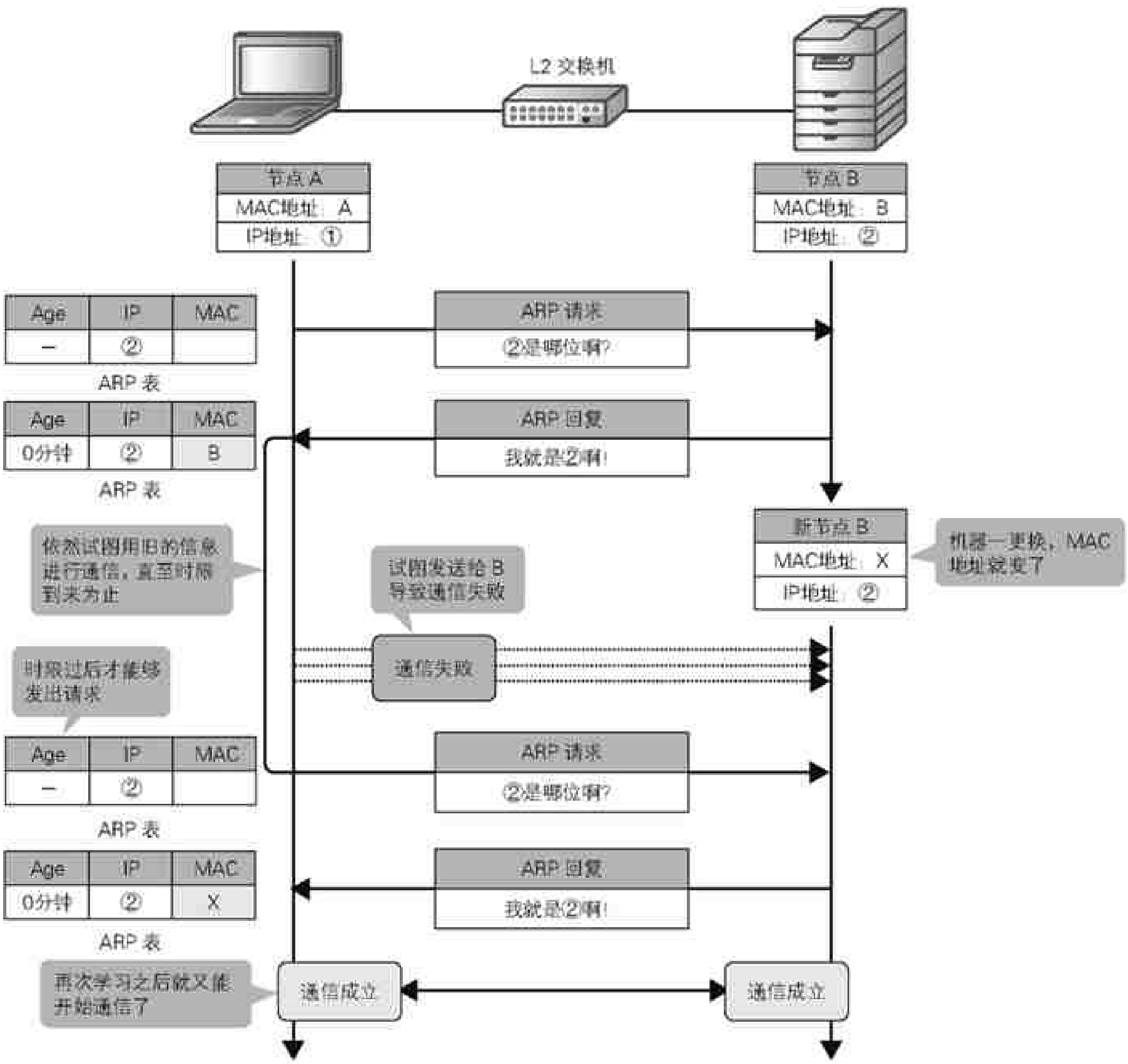


图 **2.1.39　MAC** 地址一变就无法通信了

2.1.3.2　抓取 ARP 包，观察它的写法

下面两张图是我们抓取到的 ARP 请求包和 ARP 回复包的实例。

ARP 请求为广播发送，所以目的地址为 FF-FF-FF-FF-FF-FF，此外我们还能看到数据部分写入了很多地址信息；ARP 回复为单播发送，所以目的地址为发来 ARP 请求的节点的 MAC 地址，我们能看到该数据部分也同样写入了很多地址信息。

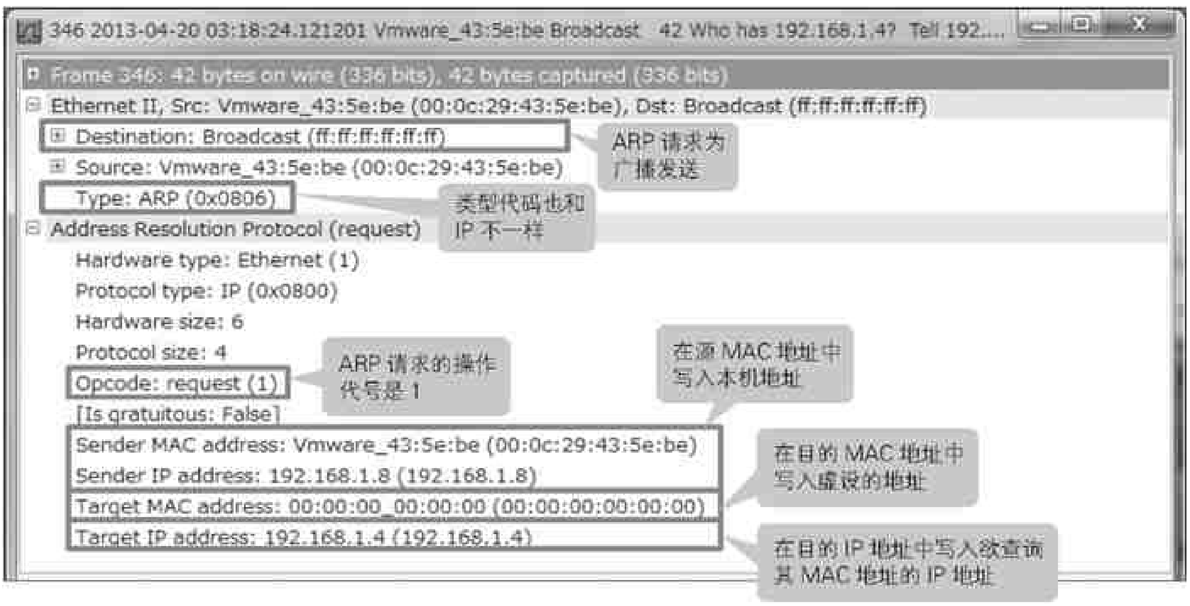


图 **2.1.40　ARP** 请求是通过广播发送的

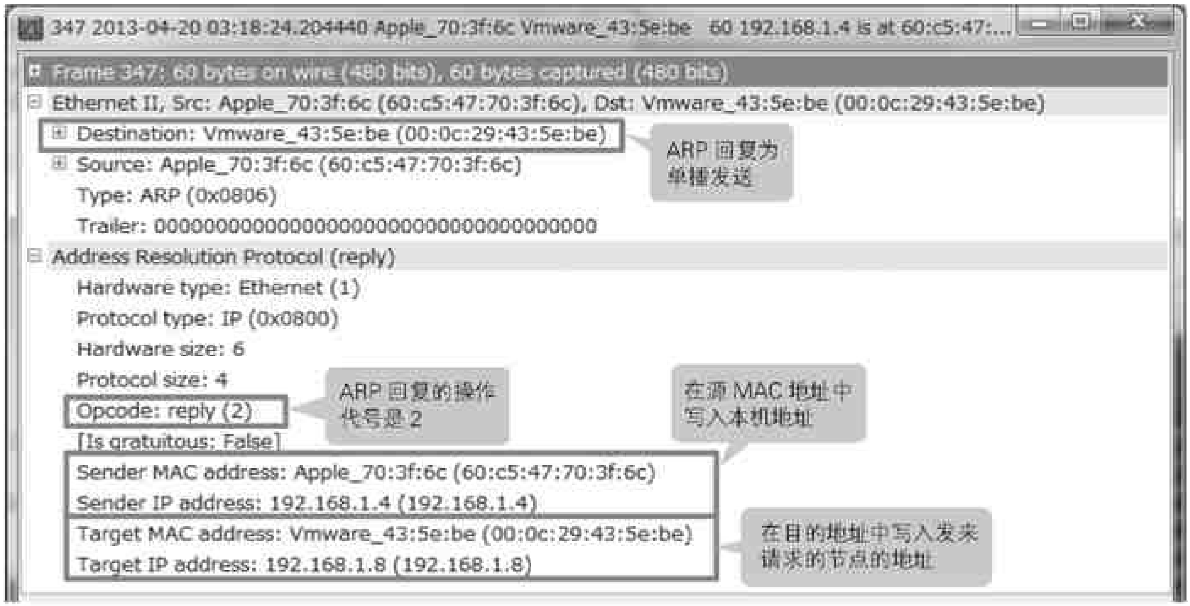


图 **2.1.41　ARP** 回复是通过单播发送的

2.1.3.3　有几个特殊的 **ARP**

ARP 包是支撑 TCP/IP 通信初期阶段的非常重要的帧，这里发生问题就会无法进行通信。为了防止出现这样的情况，人们设计了几个特殊的 ARP 用来及时更新周边的信息（MAC 地址表和 ARP 表等），它们统称为 GARP（Gratuitous ARP，无故 ARP），下面我们就来详细说明 GARP。

OS 通过 **GARP** 能在 **MAC** 地址发生变化时发出通告

GARP 是一种特殊的 ARP，它会在数据部分的源 IP 地址和目的 IP 地址中写入本机地址。GARP 肩负着双重责任，一个是检查是否存在相同的 IP 地址，另一个则是更新相邻设备的表。

- 检查是否存在相同的 **IP** 地址

  在公司或学校的网络环境中，我们可能会不经意就设置了和别人一样的 IP 地址。这时候 OS 会弹出报错信息警告我们 IP 地址冲突，提醒我们去修改 IP 地址，这个报错就是基于 GARP 的信息判断出来的。IP 地址设置好之后 OS 并不急于马上反映出来，而是发出 ARP 请求去检查周边是否存在设有相同 IP 地址的节点。如果收到 GARP，OS 就认为设有相同 IP 地址的节点确实存在，弹出报错信

息；如果没有收到 GARP，OS 才会去反映前面设置好的 IP 地址。

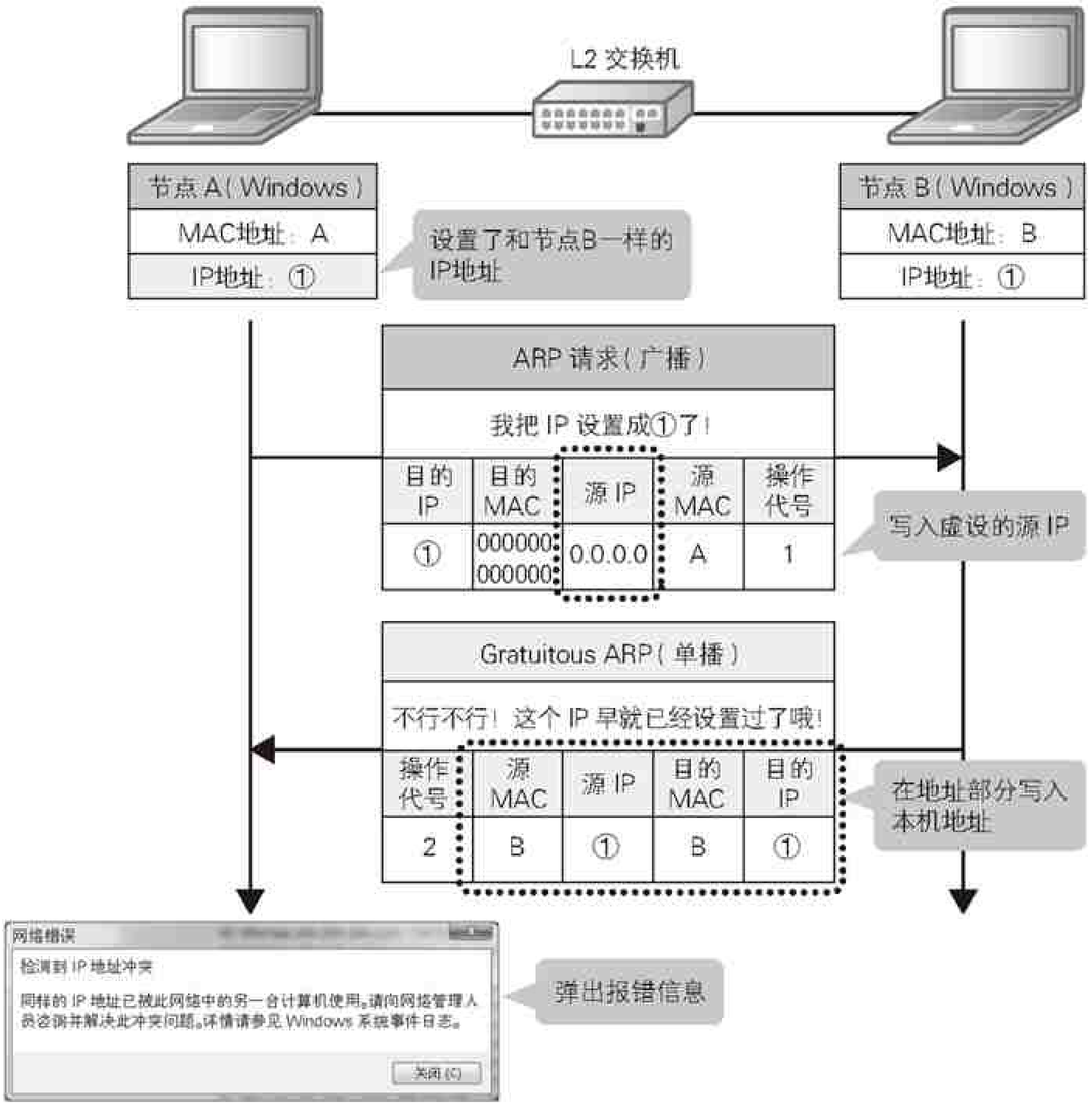


图 **2.1.42**　通过 **GARP** 检查是否存在 **IP** 地址冲突

- 更新相邻设备的表

这里所说的表指的是 ARP 表和 MAC 地址表，二者都是通过 GARP 获得更新的，不过形式稍微有些不同。

首先我们来看 GARP 是如何更新 ARP 表的。当设备因发生故障或 EoS（End of Support，终止支持）等原因需要更换成新机器的时候，旧的 MAC 地址会被新的取代。然而，周边的节点即使拥有新机器的 ARP 条目也不会去自动更新 MAC 地址，而是继续沿用旧

的信息，结果导致通信无法完成。这时候就要用到 GARP。新的机器在对接或启动时会发出 GARP 通告[3]，声明原来的 MAC 地址已变，而收到 GARP 的节点就会去更新该 ARP 条目。

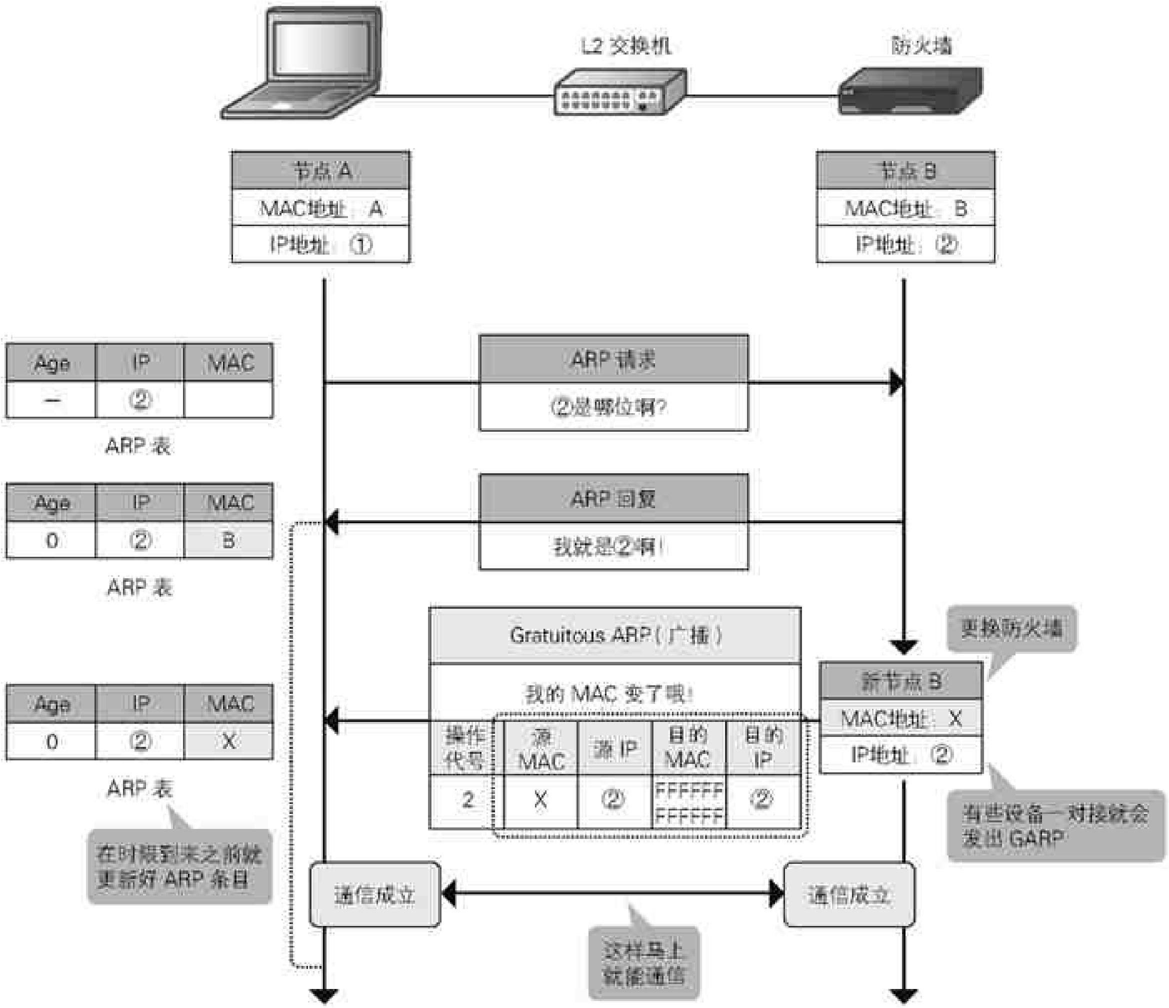


图 **2.1.43**　通过 **GARP** 更新 **ARP** 条目

接下来，我们看看 GARP 是如何更新 MAC 地址表的。

这里以虚拟环境中的 GARP 为例进行说明。虚拟软件拥有实时迁移功能，它能够在不造成停机（虚拟机）的前提下对物理服务器进行实时迁移，这在 Xen 中叫作 XenMotion，在 VMware 中叫作 vMotion。执行实时迁移时，虚拟机会从一台物理服务器移动到另一台物理服务器，然而 L2 交换机并不知道这一情况，于是虚拟机移动之后会发出 GARP[4]。L2 交换机在收到 GARP 后，会去查看该 GARP 的源 MAC 地址，并更新 MAC 地址表，使迁移之后的物理

服务器能够维持正常通信。

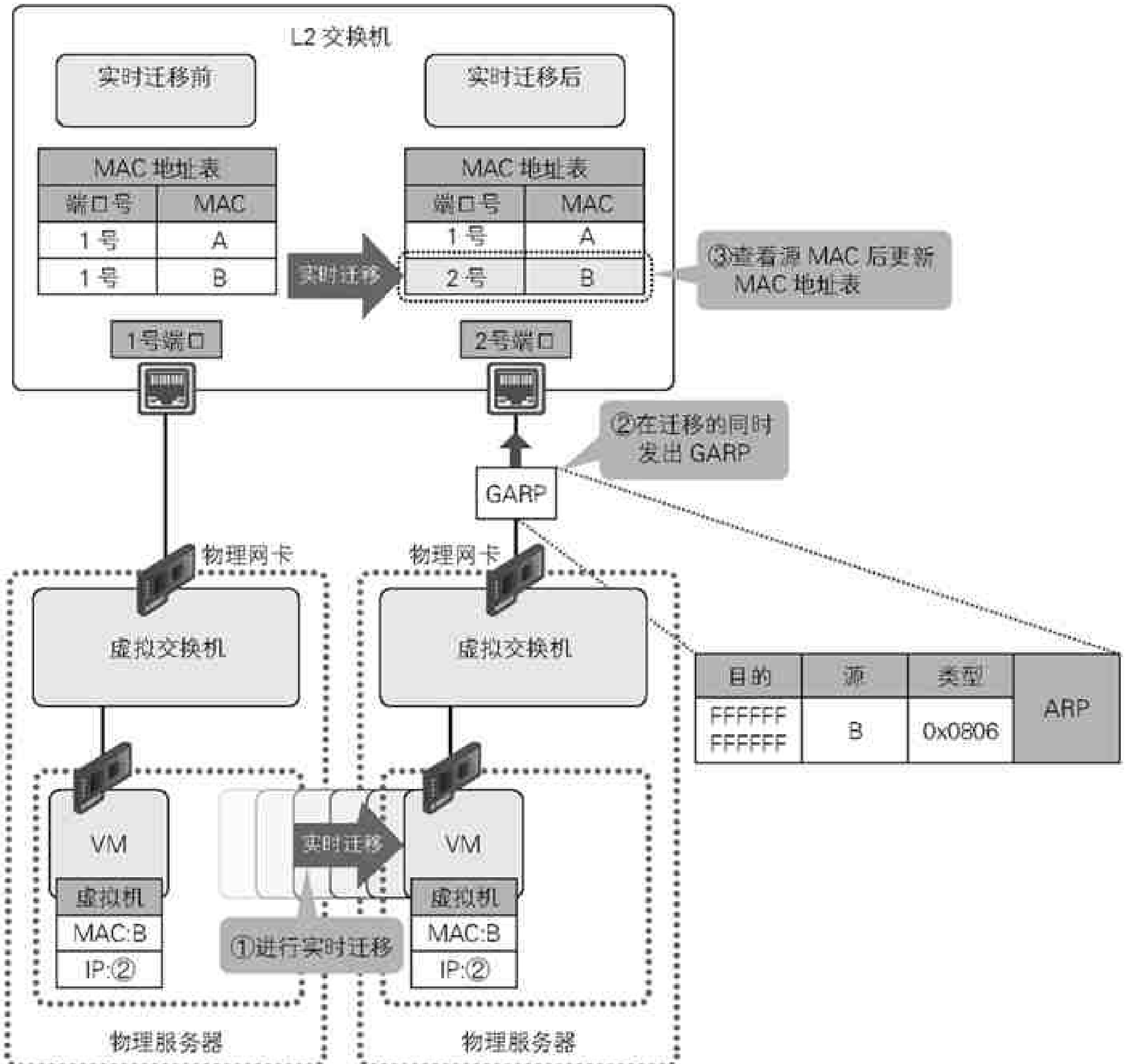


图 **2.1.44**　通过 **GARP** 追踪实时迁移之后的虚拟机

[3] 也有些设备并不发出 GARP 通告，遇到这种情况时我们需要自己去清空周边节点的 ARP 表。

[4] VMware 的 vMotion 采用的是 RARP（Reverse Address Resolution Protocol，反向地址转换协议）。协议虽然不同，原理却是一样的，RARP 也会去更新 MAC 地址表。

2.2 网络层的技术

网络层将不同的网段连接起来，确保端到端的正常通信；与此相对，数据链路层只是将同一网段中的节点连接起来而已。假如我们要连接国外的 Web 服务器，由于网段不同，在数据链路层这个层面是根本无法连接起来的，而网络层却能将数据链路层中的一个个小网段拼接成一个大网络。当今，互联网已经是我们日常生活中不可或缺的存在，“互联网”其实就是人们将“相互”（Inter）和“网络”（Network）合二为一创造出来的一个特殊词汇（不过现在已经是个普通名词了）。大量的网段彼此相连，就形成了互联网。

2.2.1 网络是由网络层拼接起来的

网络层是确保不同网段中的节点能够彼此相连的层。前面已经讲过，数据链路层是确保同一网段中的相邻节点能够彼此相连的层。网络层则负责将数据链路层中形成的网段拼接起来，使不同网段中的不相邻节点也能够互联通信。为此，网络层要为报文段添加 IP 报头。这种添加 IP 报头的处理叫作“分组化处理”（国内很多教材称作“IP 封装”），经过报文分组处理得到的数据叫作 IP 数据包。网络层中定义了分组化的各种方式。

网络层是 OSI 参考模型中从下往上数的第三层。发送信息时，网络层将来自传输层的数据（报文段）进行分组化处理后交给数据链路层（必要时还需要进行分片处理）；接收信息时，则是对来自数据链路层的帧做一个和分组以及分片恰好相反的重组处理后交给传输层。

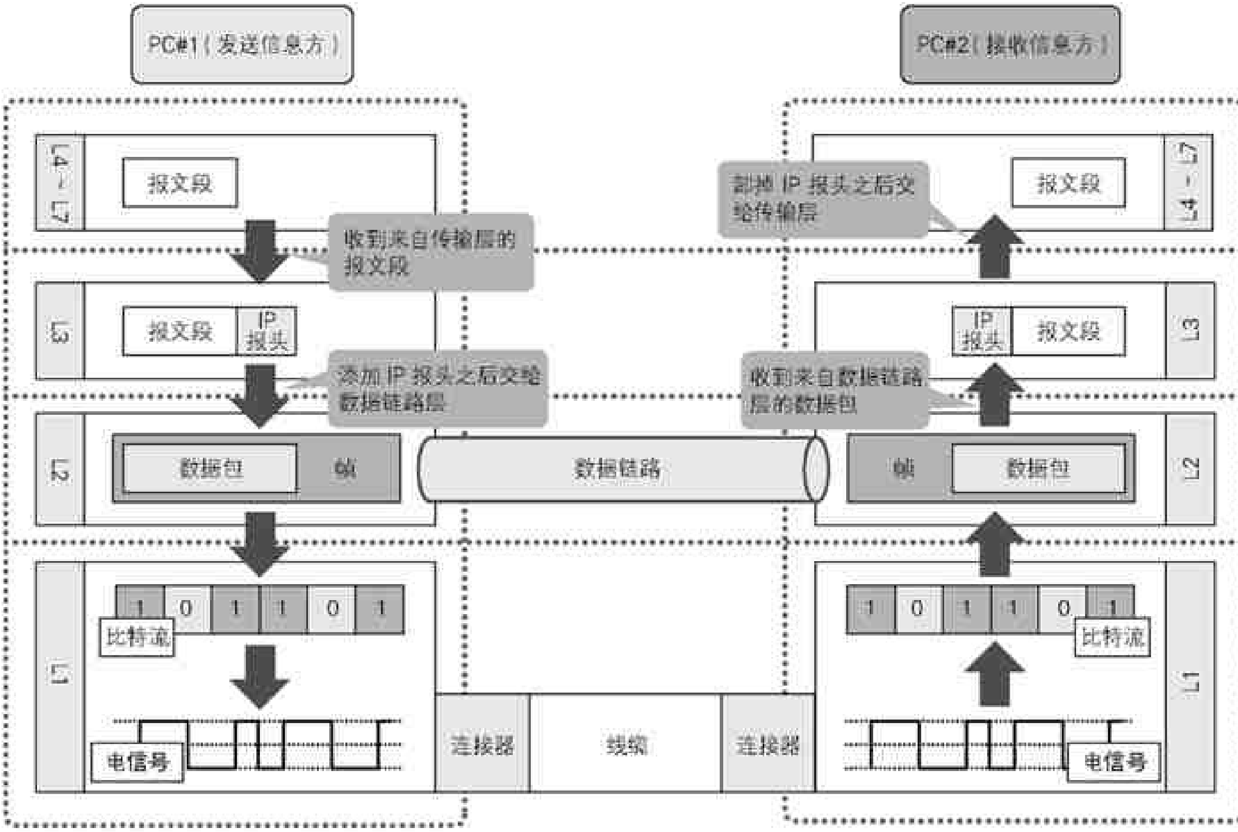


图 **2.2.1**　网络层为报文段添加 **IP** 报头

2.2.1.1　添加 **IP** 报头，进行分组化处理

在网络层的分组化处理中为报文段添加的头部叫作 IP 报头。前面介绍过的以太网报头非常简单，IP 报头就稍微复杂一些了。为了让报文段在世界各地的各种网络中通行无阻而无需在意网络差异，IP 报头中被添加进了大量的信息。

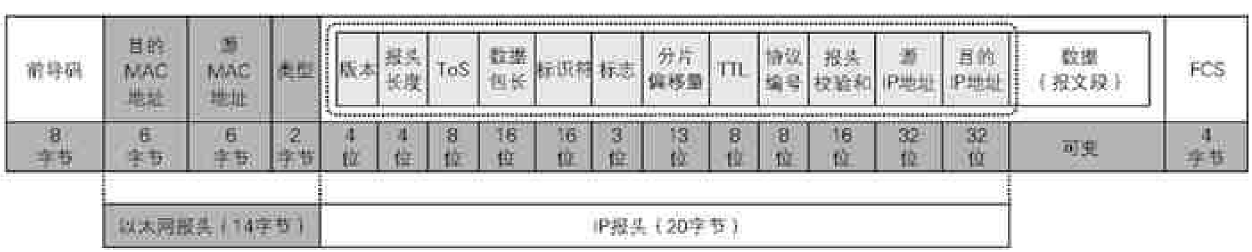


图 **2.2.2**　**IP** 报头中包含着诸多的信息（**IPv4** 报头，无可选字段）

IP 报头中包含着诸多的信息，但是我们并不需要去全盘理解。本书是围绕如何设计服务器端展开的，因此下面仅介绍一些与该内容相关的常见

字段以及经常需要我们去查询的字段。

IP 版本分 **IPv4** 和 **IPv6** 两种

首先我们来看一下版本。版本是指 IP（Internet Protocol，互联网协议）的版本，IPv4 为第四版（用二进制写是 0100），IPv6 是第六版（用二进制写是 0110）。版本不同，后面的 IP 报头中的信息就不一样。图 2.2.3 和图 2.2.4 展示的是 IPv4 的 IP 报头。

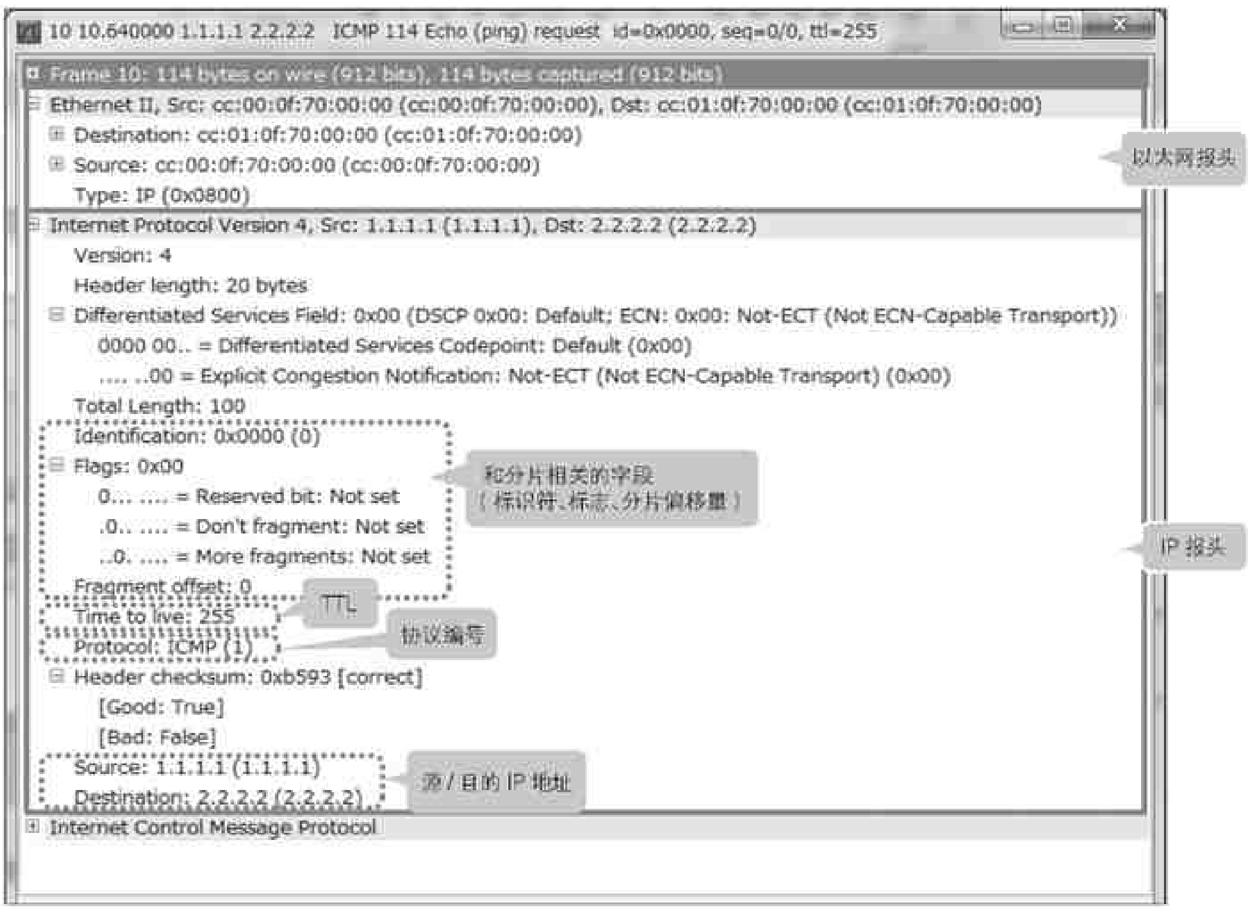


图 **2.2.3　Wireshark** 对 **IP** 报头进行分析的画面

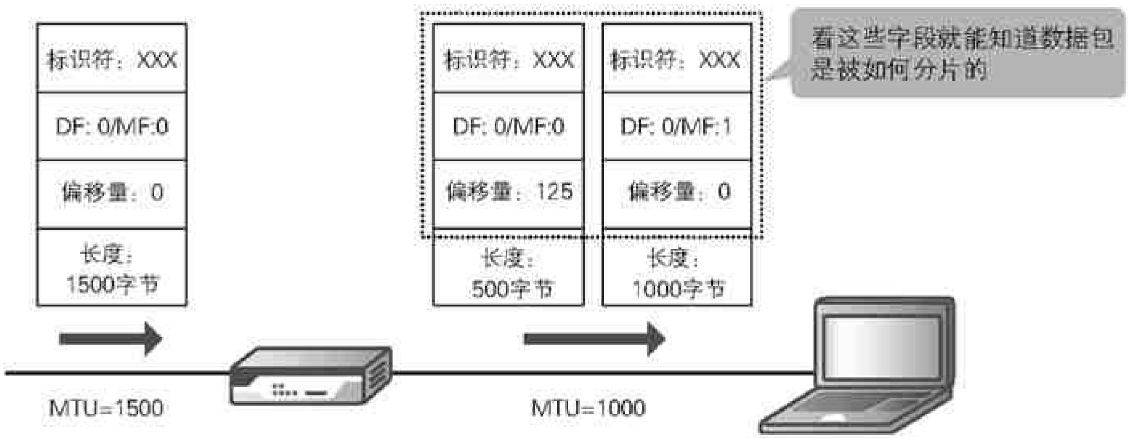


图 **2.2.4**　根据 **IP** 报头信息可将被分片的数据包还原

IPv6 是在 IPv4 地址资源日渐枯竭的背景下大张旗鼓地出现于网络界的，然而从目前的情况来看，新旧版本的过渡和交接并不尽如人意。至少对服务器端而言，IPv6 对服务器应用程序的支持较少，引进的优势也不明显，因此 IPv4 的时代可能仍将继续下去，不会出现突然需要全面更换版本的情况。不过我们仍然需要认清今后的发展趋势，逐渐开始考虑为服务器端更换 IP 版本的问题。本书对 IPv6 的说明到此为止，后面将不再触及。

标识符、标志和分片偏移量用于分片

标识符、标志和分片偏移量这三个字段都和数据包的分片（报文分片）有关，它们表示了数据包是被如何分片的。

- 标识符

  标识符是生成数据包时被随机分配的数据包 ID，长度为 16 位。当 IP 数据包因总长度超过 MTU 而被分片时，目的主机会根据这个 ID 对数据包进行重组。

- 标志

  标志的长度为 3 位。前 1 位不用；第 2 位叫作 DF 位（Don't fragment bit，禁止分片位），表示是否允许分片，0 代表允许，1 代表不允许；第 3 位叫作 MF 位（More fragment bit，后继分片位），表示分片之后的数据包是否仍有后续，0 代表没有，1 代表有。如果在 PPPoE 环境中将 MTU 设置错误数据包就会丢失，特定

的网页就会消失，这种现象和标志字段有关，详情将在 3.1.1.3 节中解说。

- 分片偏移量

  分片偏移量表示分片之后的数据包位于原始数据包的哪个位置（从原始数据包的开始处算起），以 8 字节为单位。分片之后的第一个数据包的值为 0，后面的数据包则会写入表示其位置的值。主机收到数据包后根据这些值去安排数据包的顺序。

TTL 表示数据包的寿命

TTL（Time To Live，生存时间）表示的是数据包的寿命。在 IP 的世界里，人们用数据包经过的路由器个数表示数据包的寿命，经过的路由器个数[5]叫作跳跃计数。每经过一个路由器，也就是每经过一个网段，TTL 的数值就要减一，当计数到零时该数据包就会被丢弃。丢弃数据包的路由器会返回一个 ICMP（Internet Control Message Protocol，互联网控制报文协议）数据包，该包显示了“Time-To-Live exceeded（类型 11/代号 0）”的消息，通知主机已将原数据包丢弃。关于 ICMP 的内容将在 2.2.5 节中详细说明。

> [5] 其实，数据包每经过一台运行于网络层或网络层之上的设备时数值也都要减一。例如，经过 L3 交换机或负载均衡器时 TTL 的数值也会减少。

TTL 用于防止路由循环。路由循环是指错误的路由设置导致 IP 数据包在同一个地方不断循环的现象。IP 数据包即使只是在同一个网段中不断循环，TTL 仍然会计数，导致数据包最后会被丢弃。不断循环的数据包并不会一直占用带宽。

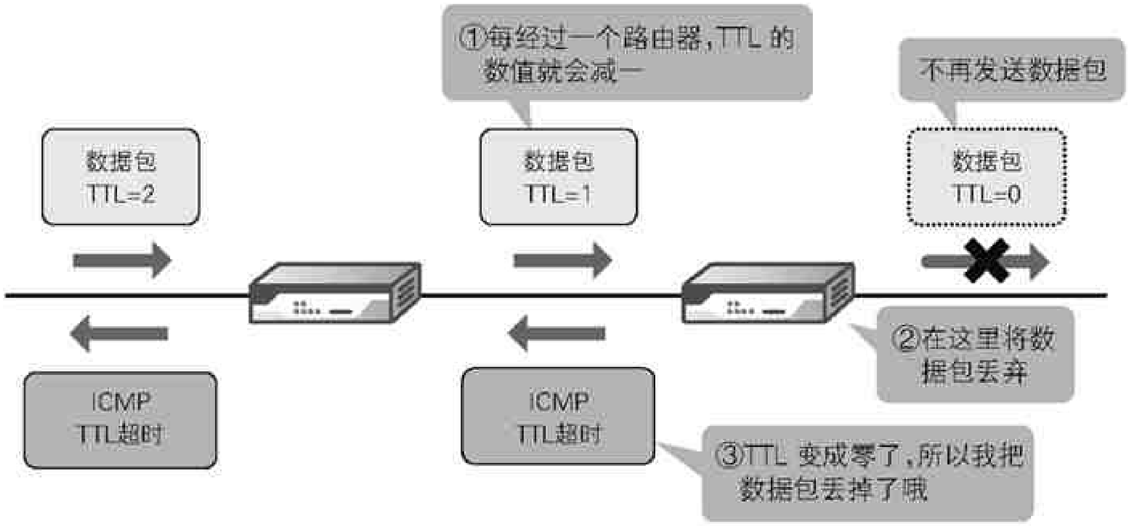


图 **2.2.5**　**TTL** 每经过一个路由器，数字就会减一

协议编号表示数据遵循的是哪种协议

协议编号表示了该数据遵循的是哪一种协议。人们定义了很多种协议，笔者将工作现场中常见的协议整理成了如下所示的一张表。

表 **2.2.1**　现场中常见的协议编号

编号	协议
1	ICMP（Internet Control Message Protocol，互联网控制报文协议）
2	IGMP（Internet Group Management Protocol，互联网组管理协议）
6	TCP（Transmission Control Protocol，传输控制协议）
17	UDP（User Datagram Protocol，用户数据报协议）
47	GRE（Generic Routing Encapsulation，通用路由封装协议）

50	ESP（Encapsulating Security Payload，封装安全载荷）
88	EIGRP（Enhanced Interior Gateway Routing Protocol，加强型内部网关路由协议）
89	OSPF（Open Shortest Path First，开放式最短路径优先）
112	VRRP（Virtual Router Redundancy Protocol，虚拟路由器冗余协议）

源 / 目的 **IP** 地址表示网络上的地址

IP 地址表示网络上的地址，长度为 32 位，一般会采用“XXX.XXX.XXX.XXX”这样的十进制形式来表示。

源 IP 地址表示发送方本机的网络地址。网络的世界都是双向沟通的，就像一方说“劳驾把这样的数据发给我吧”，另一方回“这就发给你哦”，然后这一方又说“谢谢啦”这样，互相来往并以此成立网络世界。对方看到源 IP 地址字段，就能知道应将信息返向何处。与源 IP 地址相对的是目的 IP 地址，发送方在发送信息的时候会用到这个地址。

网络层是因 IP 地址而存在的层，我们将在下一节中详细说明。

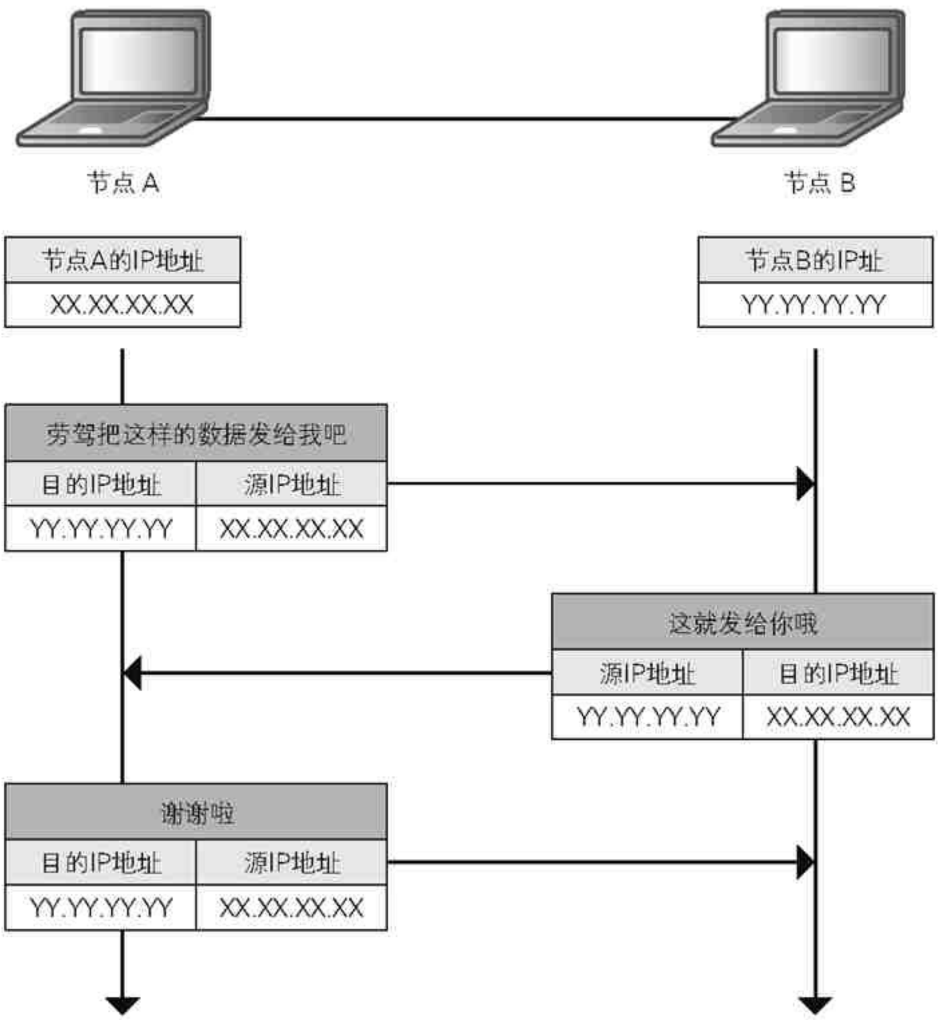


图 **2.2.6**　网络的世界都是双向沟通的

最后，我们再来把 IP 报头的字段整理成一张表看看，各位在需要快速查询时也可以参考这张表。

表 **2.2.2**　**IP** 报头中包含着诸多的信息

编号	长度	用途
版本	4位	表示IP协议的版本。 IPv4：0100（二进制） IPv6：0110（二进制）

报头长度	4位	表示IP报头的长度，以4字节为单位
ToS（Type of Service，服务类型）	8位	表示数据包的优先顺序，用于QoS（Quality of Service，服务质量）
数据包长	16位	表示整个数据包的长度，用字节单位表示
标识符	16位	用于识别数据包 分片（报文分片）时根据这个字段识别它是哪个数据包的分片。标识符由源主机随机分配
标志	3位	表示数据包是否允许分片 第 1位：不用 第 2位：表示是否可以分片 　0：允许分片 　1：不允许分片 第 3位：表示分片之后的数据包是否仍有后续 　0：没有后续 　1：仍有后续
分片偏移量	13位	表示分片之后的数据包位于原始数据包的哪个位置
TTL	8位	表示数据包的寿命 具体来说，表示数据包经过了几台路由器。默认值为64，每经过一台路由器就会减一，当路由器收到 TTL＝1的数据包时会将数据包丢弃并返回ICMP数据包
协议编号	8位	表示上层协议 ICMP ： 1 TCP ： 6 UDP ： 17
报头校验和	16位	用于检查IP报头字段是否有误的字段。由于每经过一台路由器TTL的数值就会改变，报头校验和值是以路由器为单位发生变化的

源 IP 地址	32 位	网络上的主机源地址。如果没有这个字段，对方就无法返回数据
目的 IP 地址	32 位	网络上的对方目的地址。IP 数据包将发送到这个地址

2.2.1.2 IP 地址由 32 位构成

IP 地址是一种由 32 位构成的、独一无二的识别信息，是在网络内外都有效的地址。以 IP 地址 192.168.1.1 为例，它是将 32 位按每 8 位分成 4 组，然后换算成十进制并用小数点分隔开表示的。每个区间，即每个群集，叫作 8 位字节。从前面开始，分别叫作第一 8 位字节、第二 8 位字节……依次类推。

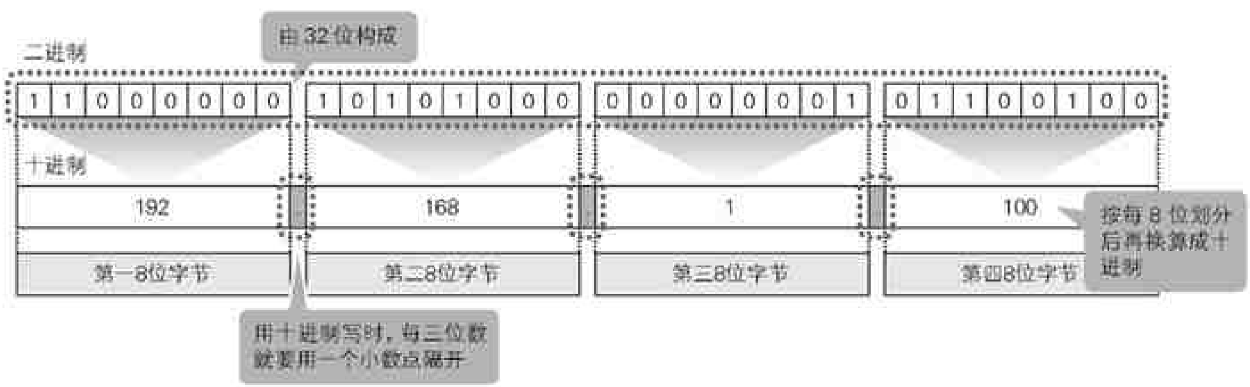


图 **2.2.7** **IPv4** 地址被小数点分隔成四个区间

子网掩码是网络和主机的标记

IP 地址并非单独存在，而是与子网掩码相结合，共同发挥作用的。IP 地址可分为两个部分，一个是网络部分，另一个则是主机部分。

网络部分表示网段本身，也就是说，它既是广播域也是 VLAN，同时还是报文段。主机部分则表示与该网段相连的节点。子网掩码就是将这两个部分分隔开的标记，子网掩码中的 1 代表网段地址，0 代表主机地址。和 IP 地址一样，设置子网掩码时每三位数（用二进制来表示的话是每八位数）就要用一个小数点隔开，最后分成四个部分并换算成十进制。举个例子，假设针对 IP 地址 172.16.1.1 设置的子网掩码是

255.255.0.0，那么我们就能知道这是网络 176.16 的主机 1.1。

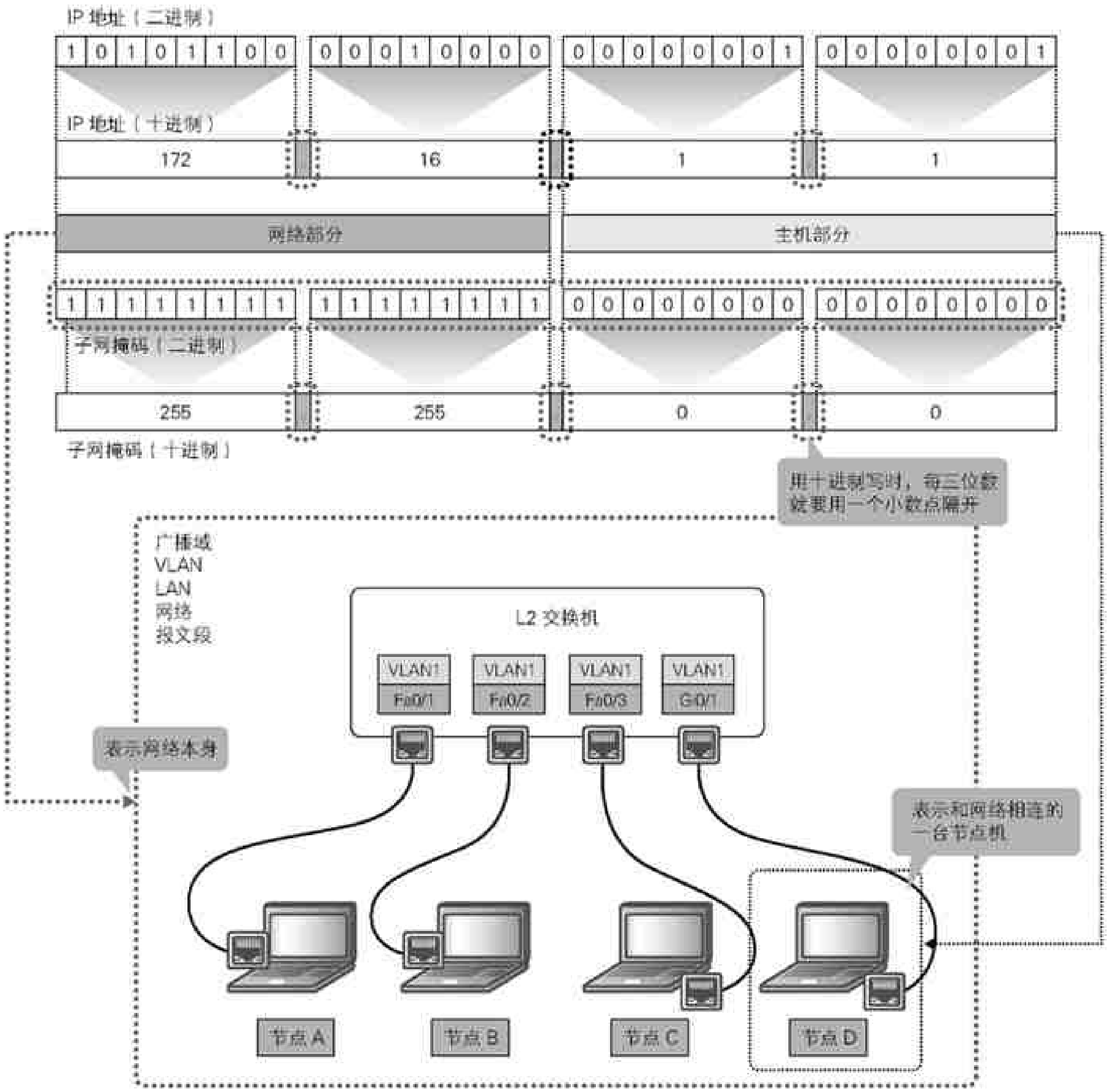


图 **2.2.8**　子网掩码将 **IP** 地址分隔开

有些特殊的 **IP** 地址已被占用，因而无法设置

是不是网络中的所有 IP 地址都能够使用呢？并不一定。有些 IP 地址是已经被系统占用了的，其中有三种常用于网络设计和故障排除，它们分别是网络地址、广播地址和环回地址。

- 网络地址

  网络地址是指主机部分的 IP 地址位都是 0 的 IP 地址，代表了网络本身。举个例子，如果针对 IP 地址 192.168.1.1 设置的子网掩码是 255.255.255.0，那么 192.168.1.0 就是网络地址。

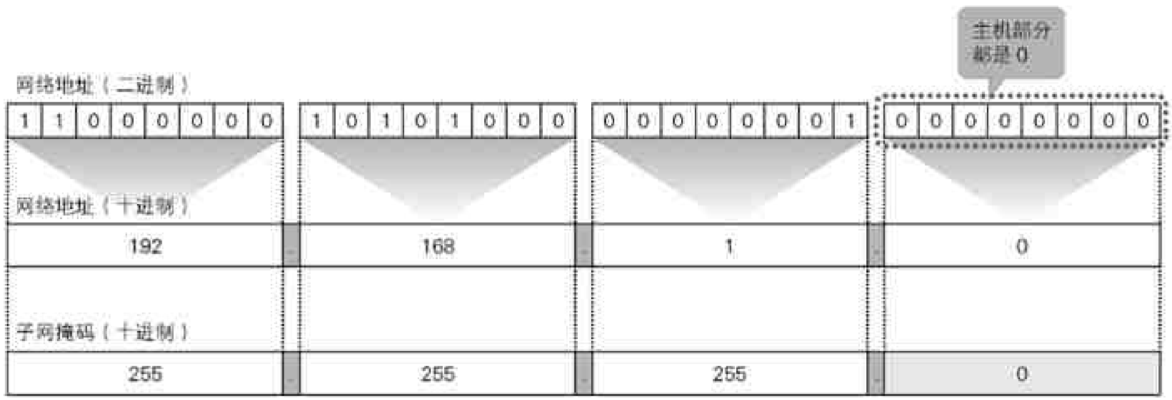


图 **2.2.9**　网络地址的主机部分都是 **0**

- 广播地址

  广播地址是指主机部分的 IP 地址位都是 1 的 IP 地址，代表了同一网段中的所有节点。举个例子，如果针对 IP 地址 192.168.1.1 设置的子网掩码是 255.255.255.0，那么 192.168.1.255 就是广播地址。

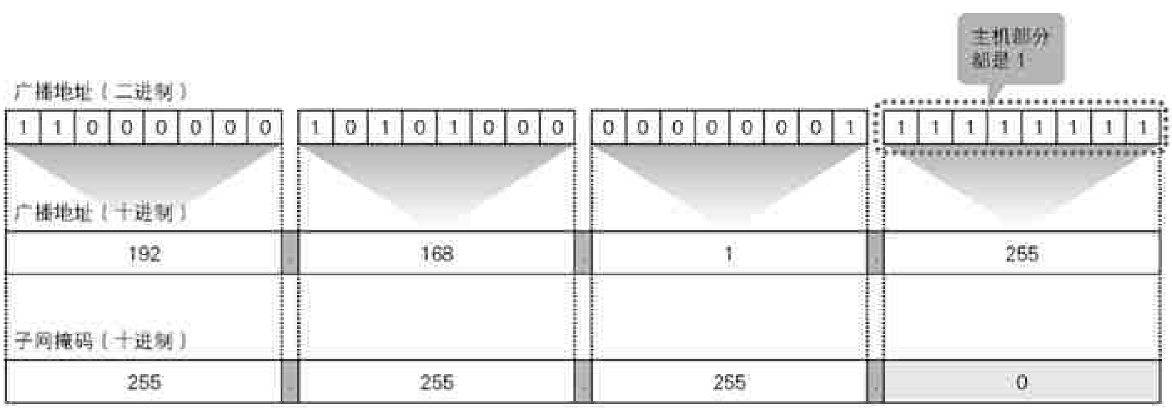


图 **2.2.10**　广播地址的主机部分都是 **1**

广播地址有三种，分别为本地广播地址、直接广播地址和有限广播地址。为了方便说明，这里我们假设某个节点设置的 IP 地址为 192.168.1.1/24。

本地广播地址指本机所属网段的广播地址。由于 192.168.1.1/24 的节点隶属于 192.168.1.0/24 网段，所以它的本地广播地址就是 192.168.1.255。向 192.168.1.255 发送报文时，同一网段中的所有节点都会收到该报文。

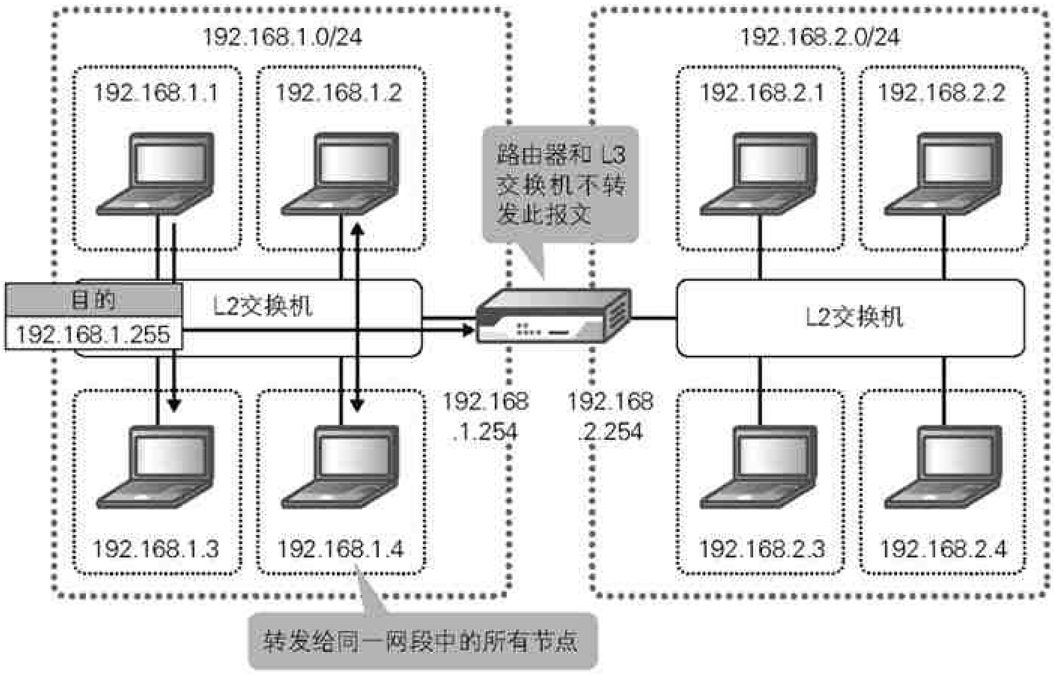


图 **2.2.11**　本地广播地址表示同一网段中的所有节点

直接广播地址是指非本机所属网段的广播地址。由于 192.168.1.1/24 的节点隶属于 192.168.1.0/24 网段，所以像 192.168.2.255、192.168.3.255 这样的地址就是非 192.168.1.255 的广播地址，也就是直接广播地址。直接广播用于远程开机（WoL，Wake-on-LAN，局域网唤醒）[6]。

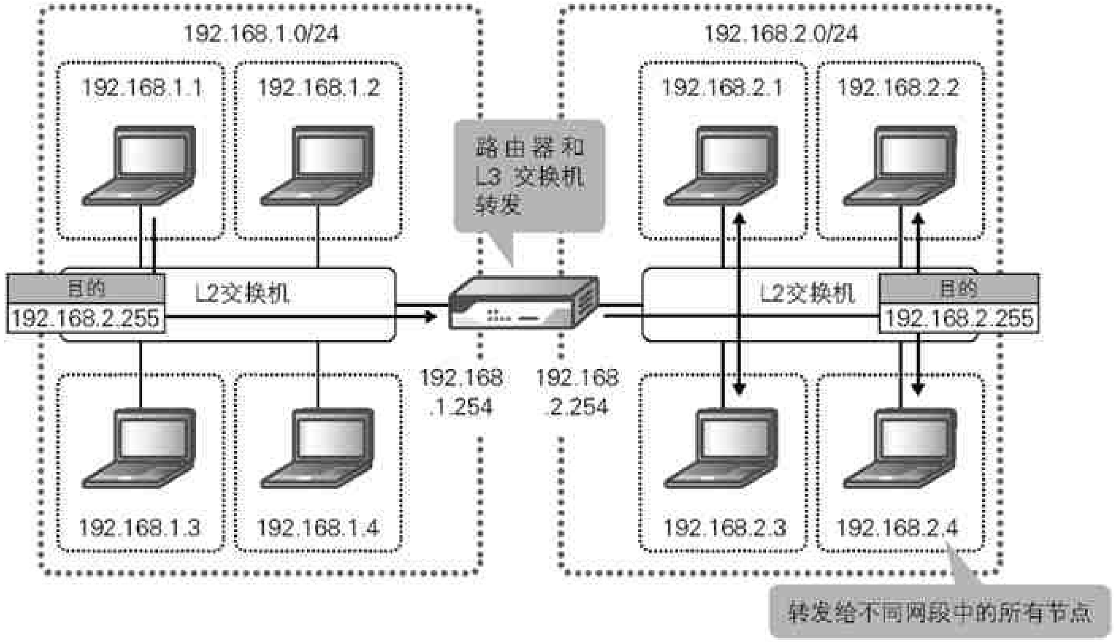


图 **2.2.12**　直接广播地址代表不同网段中的所有节点

最后我们再来看看有限广播地址。有线广播地址只有一种，其 IP 地址为 255.255.255.255。向 255.255.255.255 发送报文时，同一网段中的所有节点都会收到该报文，这一点和本地广播地址一样。不过，有限广播地址用于不知道本机 IP 地址或网络的情况，也用于 DHCP 报文的发送。关于 DHCP 的内容将在 2.2.4.2 节中详细说明。

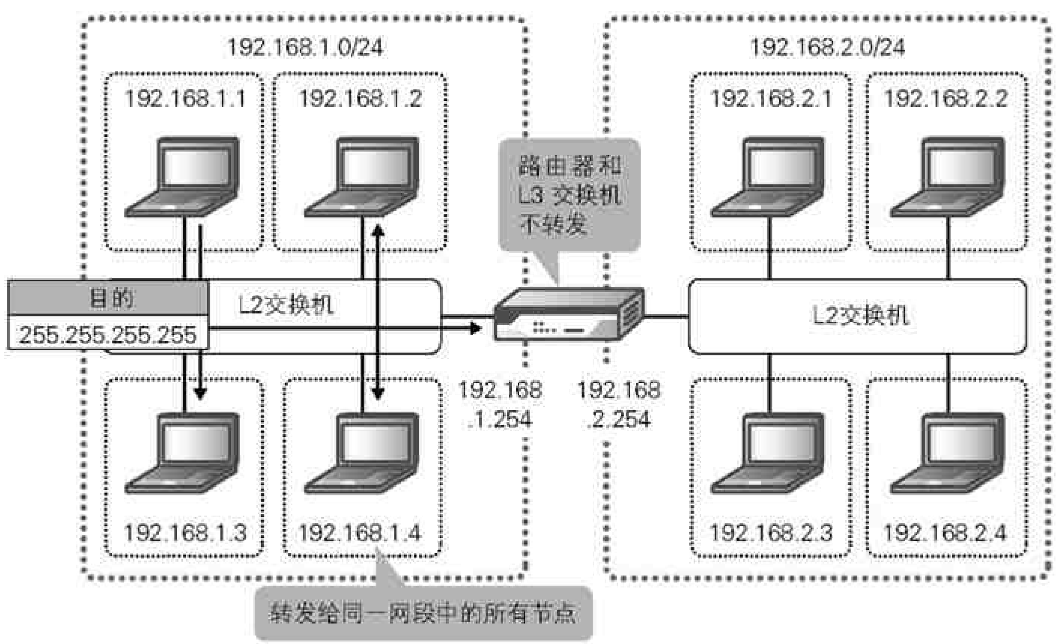


图 **2.2.13**　有限广播地址代表同一网段中的所有节点

- 环回地址

  环回地址表示设备本身的 IP 地址，它的第一 8 位字节是 127。只要第一 8 位字节是 127，后面无论是多少都没关系，但人们一般会使用 127.0.0.1/32。在 Windows 和 Mac 中，除了用户自己设置的之外，该 IP 地址都自动设为了 127.0.0.1/32。

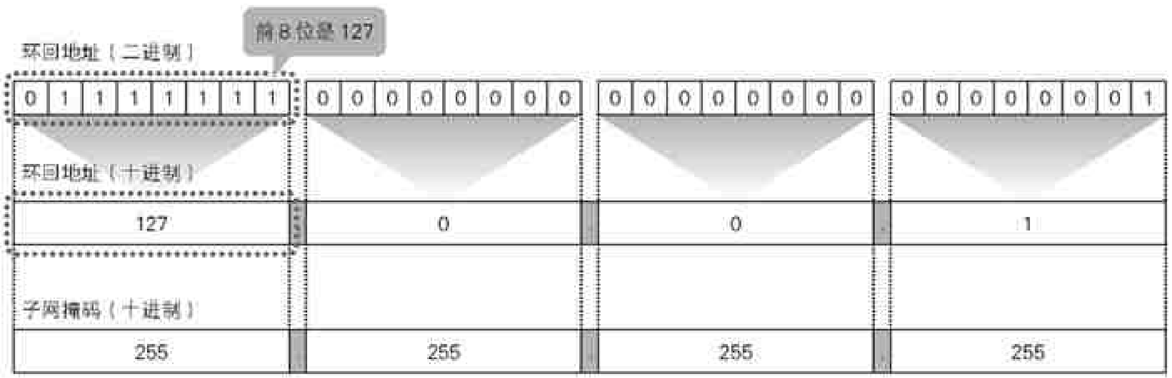


图 **2.2.14**　环回地址代表设备本身的地址

> [6] WoL 是一种通过网络远程启动 PC 或服务器的技术。它通过发送一个叫作 Magic Packet 的、由特定比特流构成的特殊报文实现开机。如果 PC 或服务器处于等待开机的状态则无法知道它们的 IP 地址，所以要使用直接广播。

分类编址，一目了然

IP 地址代表网络上的地址，每个 IP 地址在互联网上都必须是独一无二的才行，因此绝不可以随便使用。目前，由一个叫作 ICANN（The Internet Corporation for Assigned Names and Numbers，互联网名称与数字地址分配机构）的民间非营利性法人及其下属组织管理和分配着互联网上的 IP 地址。

在初期的互联网中，人们按照网络规模大致划分了 IP 地址空间，然后将 IP 地址分配给了各家企业和组织 [7] 。这里所说的网络规模叫作地址分类。

> [7] 各国隶属于 ICANN 的国家互联网注册机构负责分配自己国家的 IP 地址。

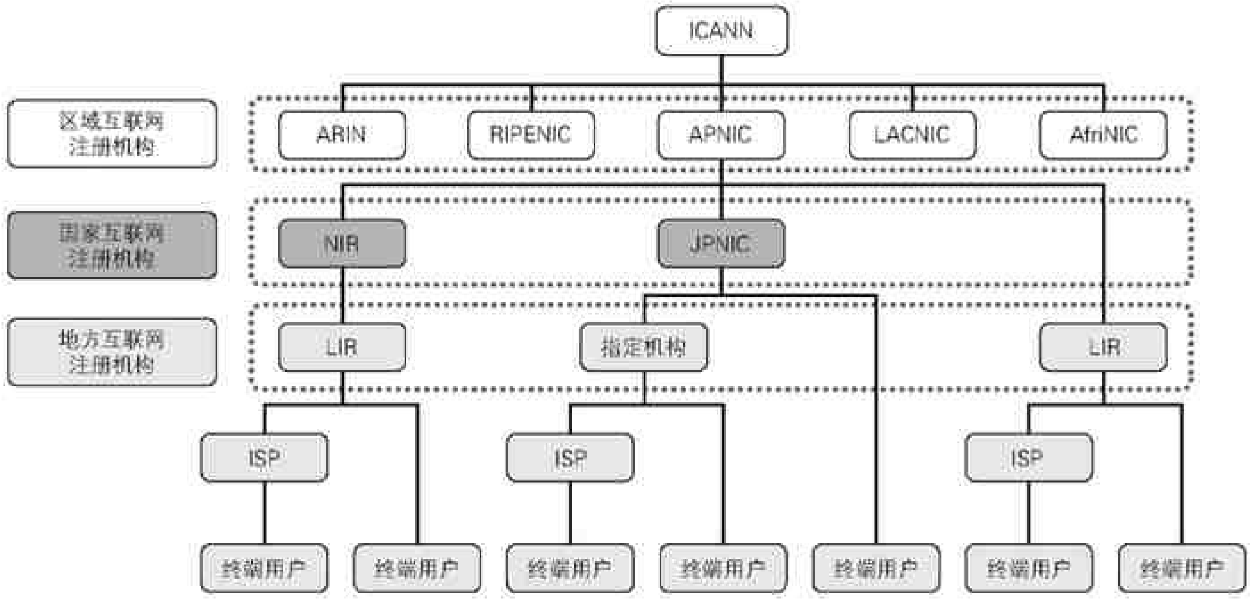


图 **2.2.15　ICANN** 及其下属机构管理着 **IP** 地址

参考 URL: https://www.nic.ad.jp/ja/ip/ipv4-pool/provider.html

IP 地址分类从 A 到 E 有五种，用前 1 ～ 4 位区别开。实际上可用于节点地址分配的是 A、B 和 C 这三种类型，网络部分和主机部分以 8 位字节为单位，也就是每隔 8 位便分隔开。这种以每 8 位分隔开的地址分配方式叫作分类编址，用这种方式分配的 IP 地址叫作分类地址。

- **A** 类

  A 类 IP 地址的第 1 位都是 0。前 8 位为网络地址，剩下的 24 位为主机地址，可分配给 16 777 214（$2^{24}-2$）台机器。拥有众多节点的 ISP（Internet Service Provider，互联网服务提供商）、在互联网创始期就参与其中的企业以及研究机关等用的就是 A 类地址。

- **B** 类

  B 类 IP 地址的前 2 位都是 10，前 16 位为网络地址，剩下的 16 位为主机地址，可分配给 65 534（$2^{16}-2$）台机器。规模比 ISP 小一些的很多大企业用的就是 B 类地址。

- **C** 类

C 类 IP 地址的前 3 位都是 110，前 24 位为网络地址，剩下的 8 位为主机地址，可分配给 254（2^8-2）台机器。所需 IP 地址比较少的中小企业等大多采用 C 类地址。

- **D** 类

D 类 IP 地址的前 4 位都是 1110，用作多播地址。多播是一对多（特定的多播组）的通信方式，比起向整个网络发送数据包（泛洪）的广播，多播的转发效率更好，因此 D 类地址一般用于流式播放等[8]。

- **E** 类

E 类 IP 地址的前 4 位都是 1111，用于研究，一般不公开使用。

[8] 有人以为 YouTube 是多播，这是误解。YouTube 服务器和浏览器进行的是一对一的 HTTP 通信，所以它是单播。

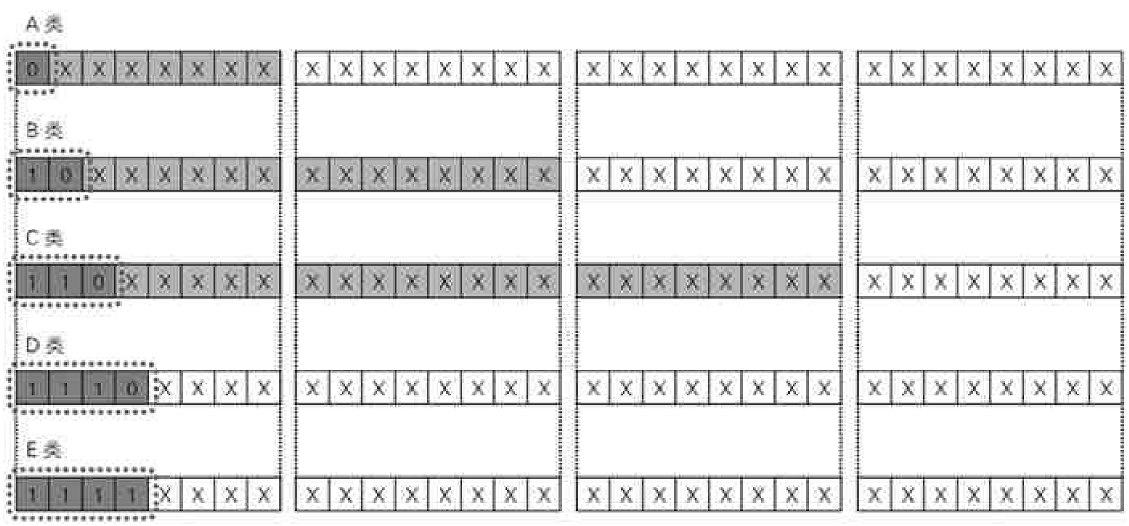


图 **2.2.16**　地址分类由前 **1** ～ **4** 位决定

表 **2.2.3**　组织的规模和所需 **IP** 地址的数量决定了使用的地址类型

类型	起点**IP**地址	终点**IP**地址	网络部分	主机部分	可分配的最大节点数[9]
A类	0.0.0.0	127.255.255.255[10]	8位	24位	16 777 214（$=2^{24}-2$）

B类	128.0.0.0	191.255.255.255	16位	16位	65 534（$=2^{16}-2$）
C类	192.0.0.0	223.255.255.255	24位	8位	254（$=2^{8}-2$）
D类	224.0.0.0	239.255.255.255	用于多播的地址		
E类	240.0.0.0	255.255.255.255 [11]	已被系统占用或用于研究		

[9] 由于网络地址和广播地址不能分配给节点，所以“－2”之后的数字为最大主机数。

[10] 0.0.0.0 和 127.0.0.0/8 已被系统占用，分别用于默认路由地址和环回地址。

[11] 255.255.255.255 已被系统占用，用于有限广播地址。

无分类编址可避免资源浪费

前面对分类编址作了详细的讲解，不过现在人们其实已经不再使用它了。为什么呢？因为分类地址按每 8 位分隔开网络和主机地址的做法虽然使人易于理解，但也带来非常多的浪费。我们以 A 类为例来说明。按该类的原则去分配，就会消耗掉一半左右可供分配的 IP 地址。试想，一家企业会需要 1600 万个 IP 地址吗？恐怕不会。分配完真正需要的 IP 地址之后，剩下的就只能闲置了，这是对资源的巨大浪费。于是人们提出了一个叫作无分类编址的全新概念。目前，ICANN 就是用这种方式去分配 IP 地址的。

无分类编址方式将分类地址分割成一种叫作子网的更小的地址单位，并以此去分配 IP 地址。无分类编址又称子网划分或 CIDR（Classless Inter-Domain Routing，无分类域间路由）。除了网络部分和主机部分之外，无分类编址中还有一个全新概念的子网部分，构成了新的网络部分。子网部分原是属于主机部分的，但现在人们利用它将地址分割成更小的单位。

下面我们以 192.168.1.0 为例，看看如何将原网络划分成若干个子网。192.168.1.1 是 C 等级地址，因此网络部分为 24 位，主机部分为 8 位。我们要在主机部分分出子网，给子网分配多少位取决于我们实际所需的 IP 地址数量和网络数量。假设需要 16 个子网，那么我们就应分配 4 位给子网以形成新的网络部分，这样才能分割出 16（2^4 ）个子网来。这

样做的话，我们能得到 IP 地址为 192.168.1.0/28 ～ 192.168.1.240/28 范围内的 16 个子网，并且对每个子网都可以分配 14（2^4 −2）个 IP 地址。

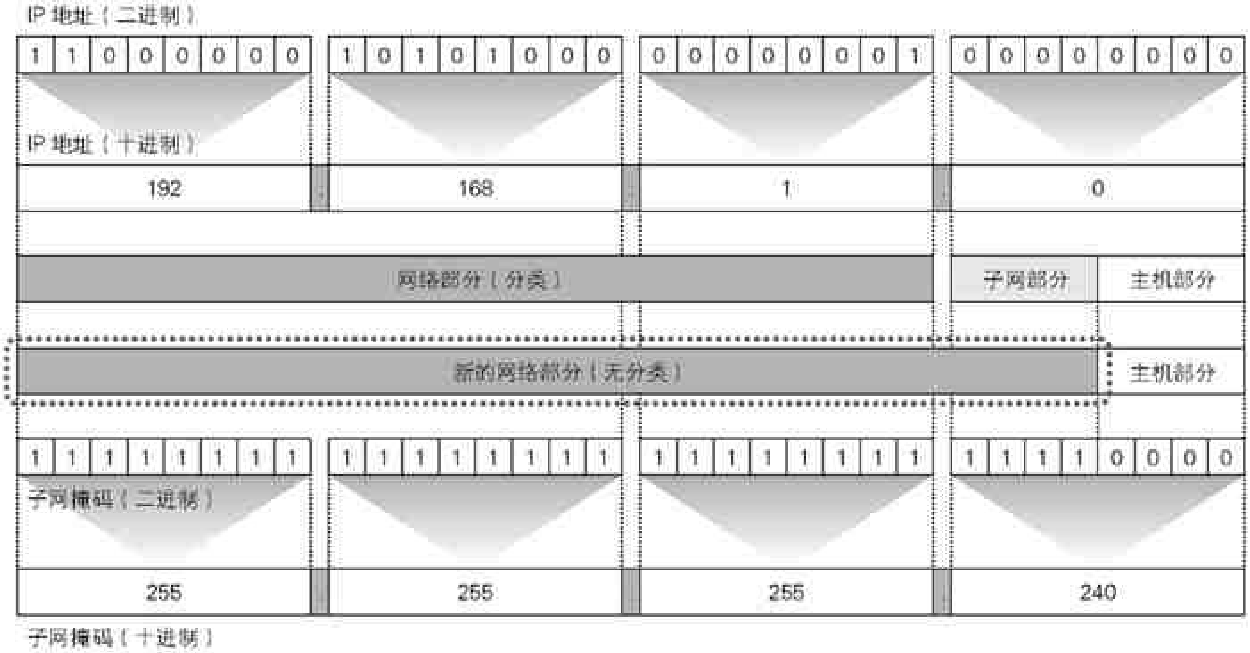


图 **2.2.17**　无分类编址将网络划分成若干个子网

无分类编址的位计算很麻烦，容易算错、也容易忘记。尤其在设计网络时，由于子网掩码不止一个，算法会更加复杂。笔者设计时一般会整理出一张表用来随时参考。表 2.2.4 就是将 192.168.1.0/24 划分成 16 个子网时整理的示例。

表 **2.2.4**　将子网整理成表就一目了然了

<table>
<tr><td>十进制写法</td><td>255.255.255.0</td><td>255.255.255.128</td><td>255.255.255.192</td><td>255.255.255.224</td><td>255.255.255.240</td></tr>
<tr><td>斜线写法</td><td>/24</td><td>/25</td><td>/26</td><td>/27</td><td>/28</td></tr>
<tr><td>最大 IP 数</td><td>254（=256-2）</td><td>126（=128-2）</td><td>62（=64-2）</td><td>30（=32-2）</td><td>14（=16-2）</td></tr>
<tr><td rowspan="16">分配网络</td><td rowspan="16">192.168.1.0</td><td rowspan="8">192.168.1.0</td><td rowspan="4">192.168.1.0</td><td rowspan="2">192.168.1.0</td><td>192.168.1.0</td></tr>
<tr><td>192.168.1.16</td></tr>
<tr><td rowspan="2">192.168.1.32</td><td>192.168.1.32</td></tr>
<tr><td>192.168.1.48</td></tr>
<tr><td rowspan="4">192.168.1.64</td><td rowspan="2">192.168.1.64</td><td>192.168.1.64</td></tr>
<tr><td>192.168.1.80</td></tr>
<tr><td rowspan="2">192.168.1.96</td><td>192.168.1.96</td></tr>
<tr><td>192.168.1.112</td></tr>
<tr><td rowspan="8">192.168.1.128</td><td rowspan="4">192.168.1.128</td><td rowspan="2">192.168.1.128</td><td>192.168.1.128</td></tr>
<tr><td>192.168.1.144</td></tr>
<tr><td rowspan="2">192.168.1.160</td><td>192.168.1.160</td></tr>
<tr><td>192.168.1.176</td></tr>
<tr><td rowspan="4">192.168.1.192</td><td rowspan="2">192.168.1.192</td><td>192.168.1.192</td></tr>
<tr><td>192.168.1.208</td></tr>
<tr><td rowspan="2">192.168.1.224</td><td>192.168.1.224</td></tr>
<tr><td>192.168.1.240</td></tr>
</table>

2.2.2 将网段连接起来

在网络层运作的设备只有路由器和 L3 交换机这两种，掌握了它们，基本上就能应对所有的网络环境。我们在前面已经说过，L2 交换机的作用是根据数据链路层（L2）中的 MAC 地址信息转发帧。与此相似，路由器和 L3 交换机是根据网络层（L3）中的 IP 地址信息转发数据包。对数据包的转发目的地进行切换的过程叫作路由选择。

2.2.2.1 利用 IP 地址进行路由选择

接下来，我们来看看路由器和 L3 交换机是如何进行路由选择的。

前面已经说过，L2 交换机是通过 MAC 地址表来交换帧的。与此相似，L3 交换机和路由器是通过路由表对 IP 数据包进行路由选择的。路由表由“目的网段”“下一跳”“路由协议”“度量值”等多种信息构成，它告诉我

们将数据包转发给哪个 IP 地址就能抵达目的网段。其中，最重要的信息是“目的网段”和“下一跳”，我们就先用这两项来说明路由选择的基本原理。

使用路由表进行路由选择

假设这里有如下设置的两个节点，即节点 A 和节点 B，它们是双向通信的，我们来看看数据包是被如何搬运的。

表 **2.2.5**　节点 **A** 和节点 **B** 的设置

节点	**IP** 地址	子网掩码	默认网关
节点 A	1.1.1.1	255.255.255.0	R1
节点 B	3.3.3.3	255.255.255.0	R2

1 →　节点 A 生成 IP 数据包，封装成帧后传递给线缆，目的是发给节点 B，这里还只是在进行单播发送。源 IP 地址是节点 A 的 IP 地址（1.1.1.1），目的 IP 地址是节点 B 的 IP 地址（3.3.3.3）。由于节点 A 无法和不同网段中的节点 B 直接通信，因此数据包将会被发给预先设置好的默认网关 R1。

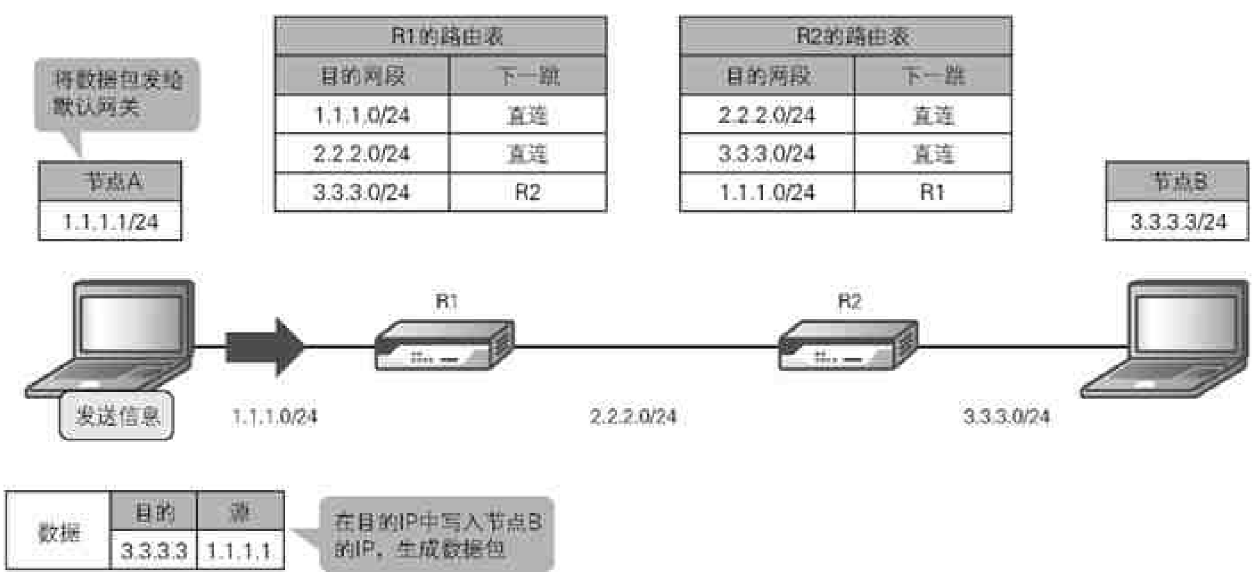


图 **2.2.18**　生成数据包并发给默认网关

2 →　收到数据包后，R1 去查看其中的目的 IP 地址并在路由表中查找。收到的数据包的目的 IP 地址是 3.3.3.3，查找后发现路由表中有 3.3.3.0/24 项，它对应的下一跳是 R2，于是数据包被发往 R2。如果 R1 在路由表中找不到该目的网段，那么数据包就会被丢弃。

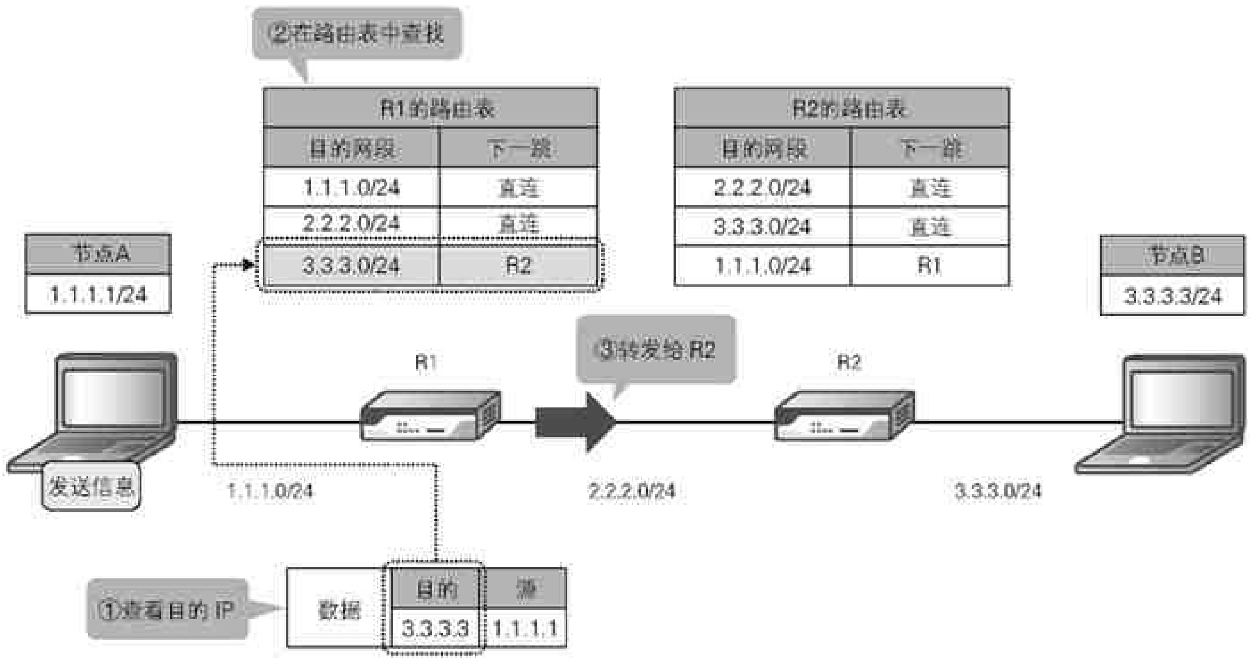


图 **2.2.19**　查看路由表，找出应将数据包发往何处的信息

3 →　收到数据包后，R2 去查看其中的目的 IP 地址并在路由表中查找。收到的数据包的目的 IP 地址是 3.3.3.3，查找后发现路由表中有 3.3.3.0/24 项，它是直连的网络，于是 ARP 去查询节点 B 的 MAC 地址，然后将数据包发给节点 B。

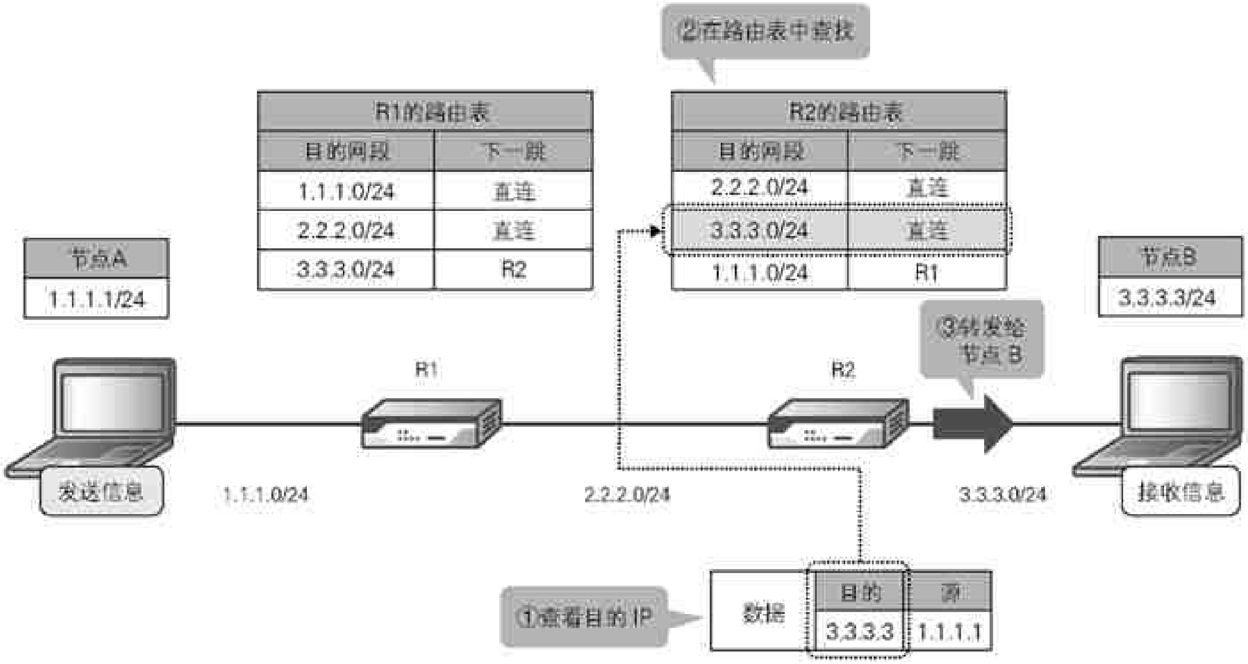


图 **2.2.20**　向同一网段中的节点发送数据包

4 →　收到数据包后，节点 B 断定该数据包是发给自己的，于是生成一个回应数据包，封装成帧后传递给线缆，目的是发给节点 A。这时候，源 IP 地址是节点 B 的 IP 地址（3.3.3.3），目的 IP 地址是节点 A 的 IP 地址（1.1.1.1）。从节点 B 的角度来看节点 A 是不同网段中的节点，因此回应数据包将会被发给预先设置好的默认网关 R2。

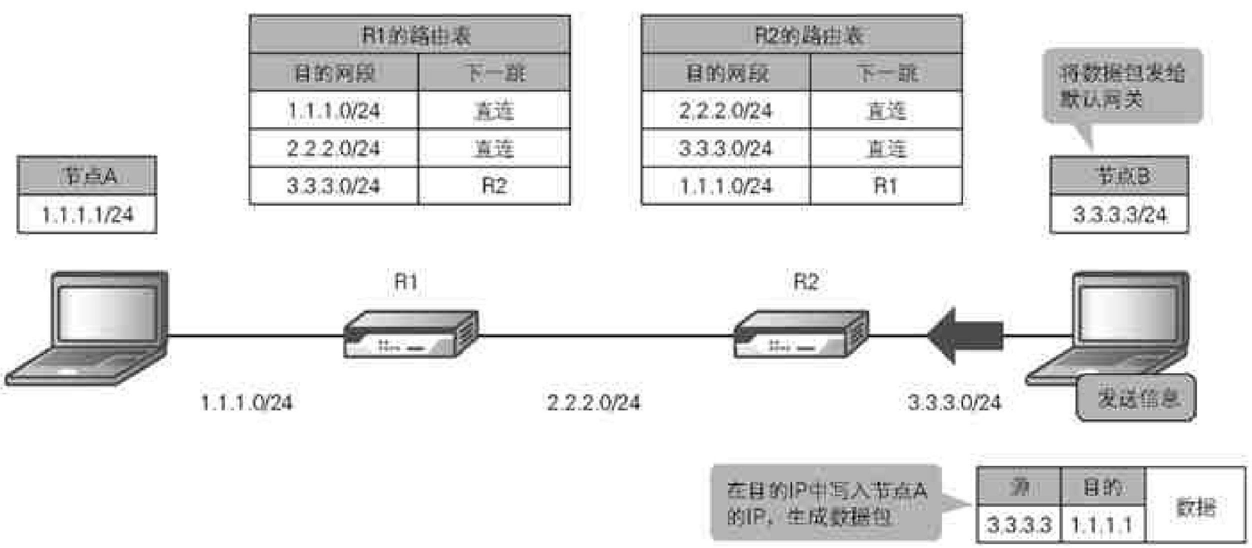


图 **2.2.21**　生成回应数据包

5 →　对于回应的数据包，R2 同样也会查看目的 IP 地址并在路由表中查找，然后转发给下一跳。在不断重复这些步骤的过程中最终将数据包发至节点 A，完成双向通信。

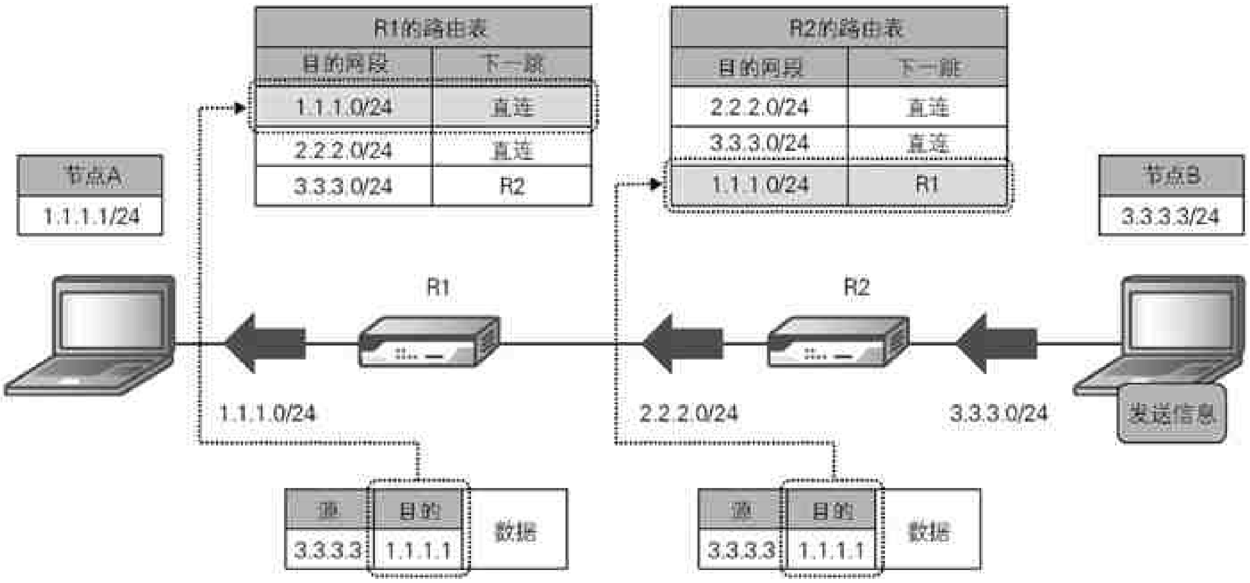


图 **2.2.22**　回应的数据包也会去查看路由表

了解路由表的构造

下面，我们来看看下图中所示的 R1 路由表。

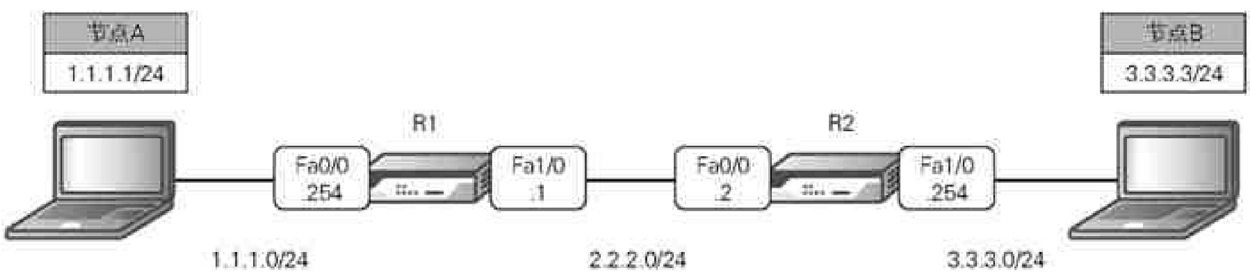


图 **2.2.23**　方便了解路由表的通信结构示例

这里我们使用的是思科路由器。思科路由器可通过命令 show ip route 查询路由表（如图 2.2.24）。各网络代表目的网段，via 后面的 IP 地址就是下一跳的 IP 地址。

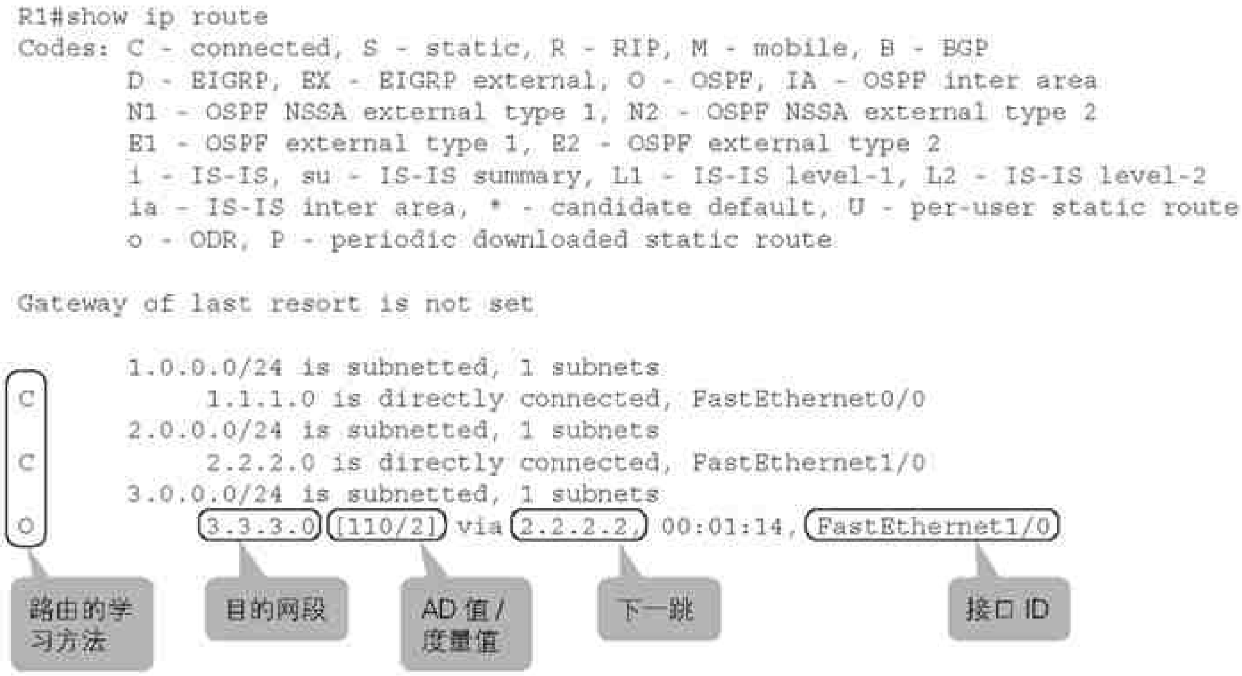


图 **2.2.24**　了解路由表的构造

顺带也介绍一下其他几个要素。最前面的 1 ～ 2 个字母表示路由的学习方法，C 代表直连，O 代表已经通过动态路由选择 OSPF 学习过，110/2 分别代表 AD（管理距离）值和度量值，这几个将在下一节中详细说明。最后还有一个接口 ID，它表示下一跳位于哪个接口。

MAC 地址和 **IP** 地址是彼此协作、共同发挥作用的

MAC 地址和 IP 地址通过 ARP 彼此协作、共同发挥作用。MAC 地址是物理地址，仅在同一网段中有效，因此，每当需要跨越网段——也就是需要跨越路由器时——都必须更换 MAC 地址才行，ARP 能解决这个目的 MAC 地址的更换问题。与此相对，IP 地址是逻辑地址，能够跨网段使用，因此，从源节点到目的节点始终保持不变。

下图展示了这两个地址是如何彼此协作、共同发挥作用的。为了帮助大家理解，图中添加了目的 / 源 MAC 地址。

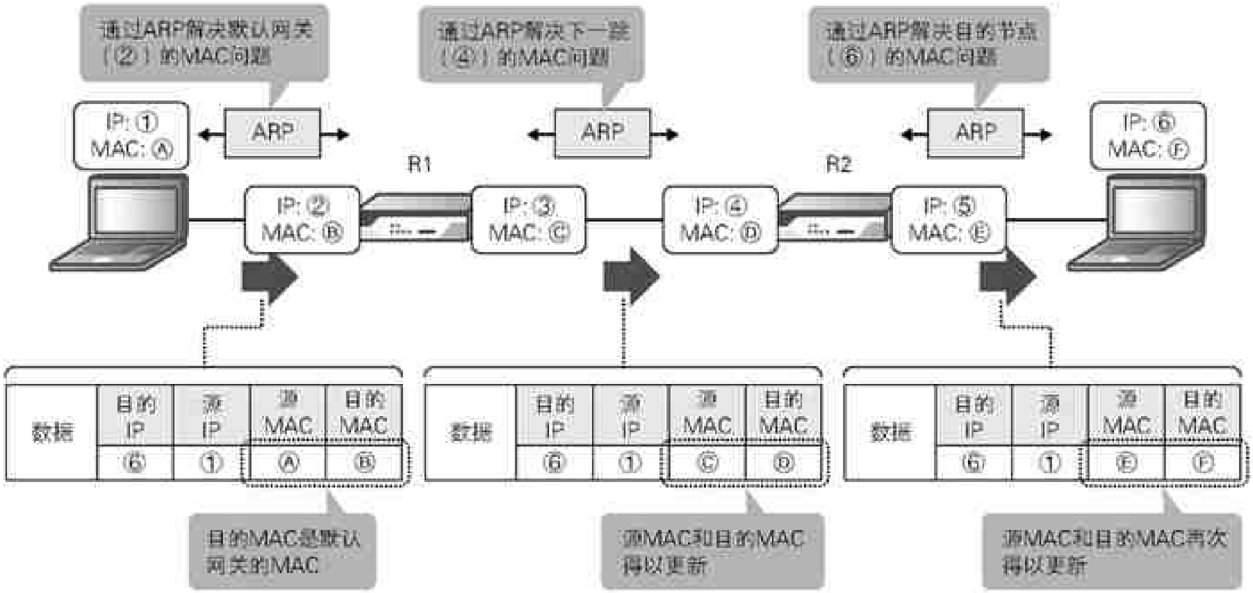


图 **2.2.25**　每跨一个网段 **MAC** 地址就要变换一次

2.2.2.2　建立路由表

路由表决定着路由选择，而如何建立这个路由表就是网络层的关键所在。下面本书将为你介绍一些入门级别的内容。

路由表有两种建立方式，一种是静态路由选择，另一种是动态路由选择。

静态路由选择

静态路由选择是手动建立路由表的方式，需要我们一个个地去设置目的网段和下一跳。这种方式容易理解也便于管理，所以比较适合小规模的网络环境，但由于需要逐一手动设置所有路由器的目的网段，所以并不适合大规模的网络环境。

以下图中的结构为例，在这种结构中，我们需要通过静态选路方式对 R1 和 R2 分别设置路由 3.3.3.0/24 和 1.1.1.2/24。如果使用的是思科路由器，用命令 ip route <network> <subnetmask> <next hop> 即可实现静态路由设置。

动态路由选择

动态路由选择是相邻的路由器彼此交换路由信息，自动建立路由表的方式。交换路由信息的协议叫作路由协议。如果是网络环境规模较大或者环境结构容易发生变化的情况，使用动态路由选择比较合适。使用这种方式，即使在新增网段时也无需设置所有的路由器，管理起来非常轻松。而且，即使中途发生故障也能自动查找迂回路径，耐故障能力较高。

不过，动态路由选择并不是万能的。经验不足的管理人员在匆忙随意的设置中一旦出错，影响就会波及整个网络，造成不可收拾的后果。为了避免出现这样的局面，动手设置之前我们一定要认真仔细地做好设计。

下面，同样还是针对前面那个网络构造，我们来看看动态路由选择是如何运作的。如图所示，R1 和 R2 彼此交换路由信息，将获得的信息添加到路由表中去。

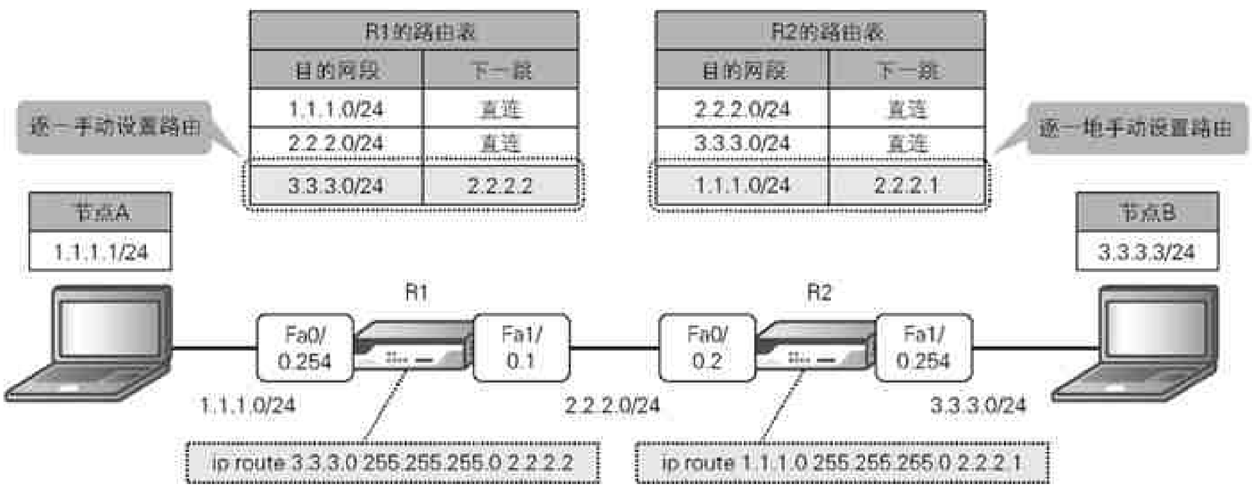


图 **2.2.26**　逐一手动设置路由

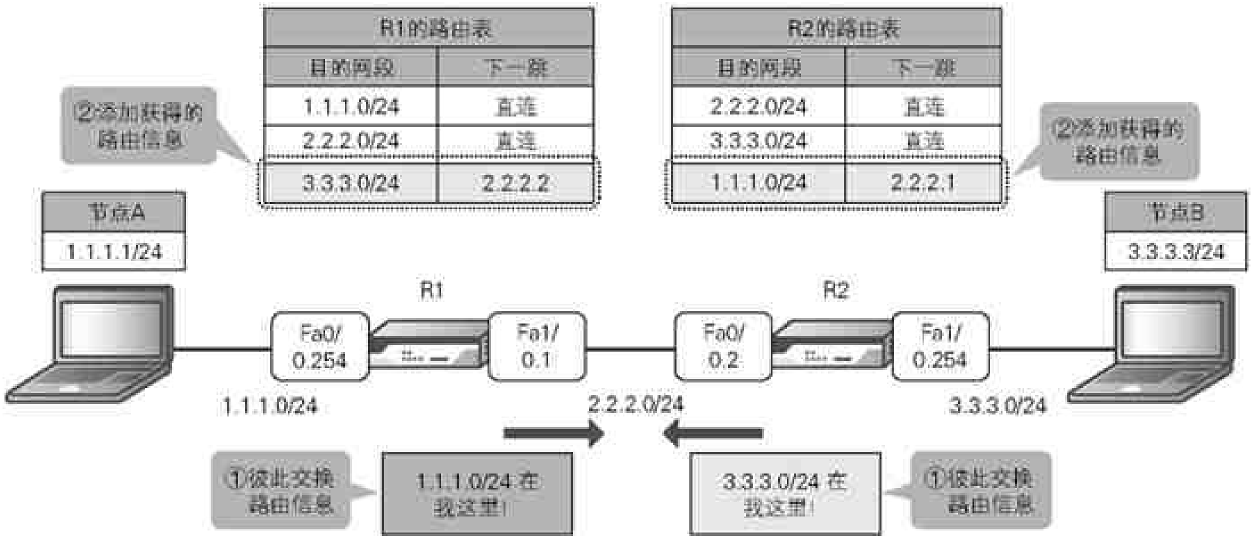


图 **2.2.27**　相邻设备彼此交换路由信息，自动建立路由表

假设我们要在这个环境中增加新的网段。新的路由器一来，现有路由器就会和新路由器交换彼此的路由信息，路由表获得全面更新（自动完成）。采用动态路由选择方式时，路由器的设置将全部由路由协议代劳，我们无需逐一手动设置。网络上的路由器能够识别出所有路径的状态叫作收敛状态，为达到收敛状态所需的时间叫作收敛时间。

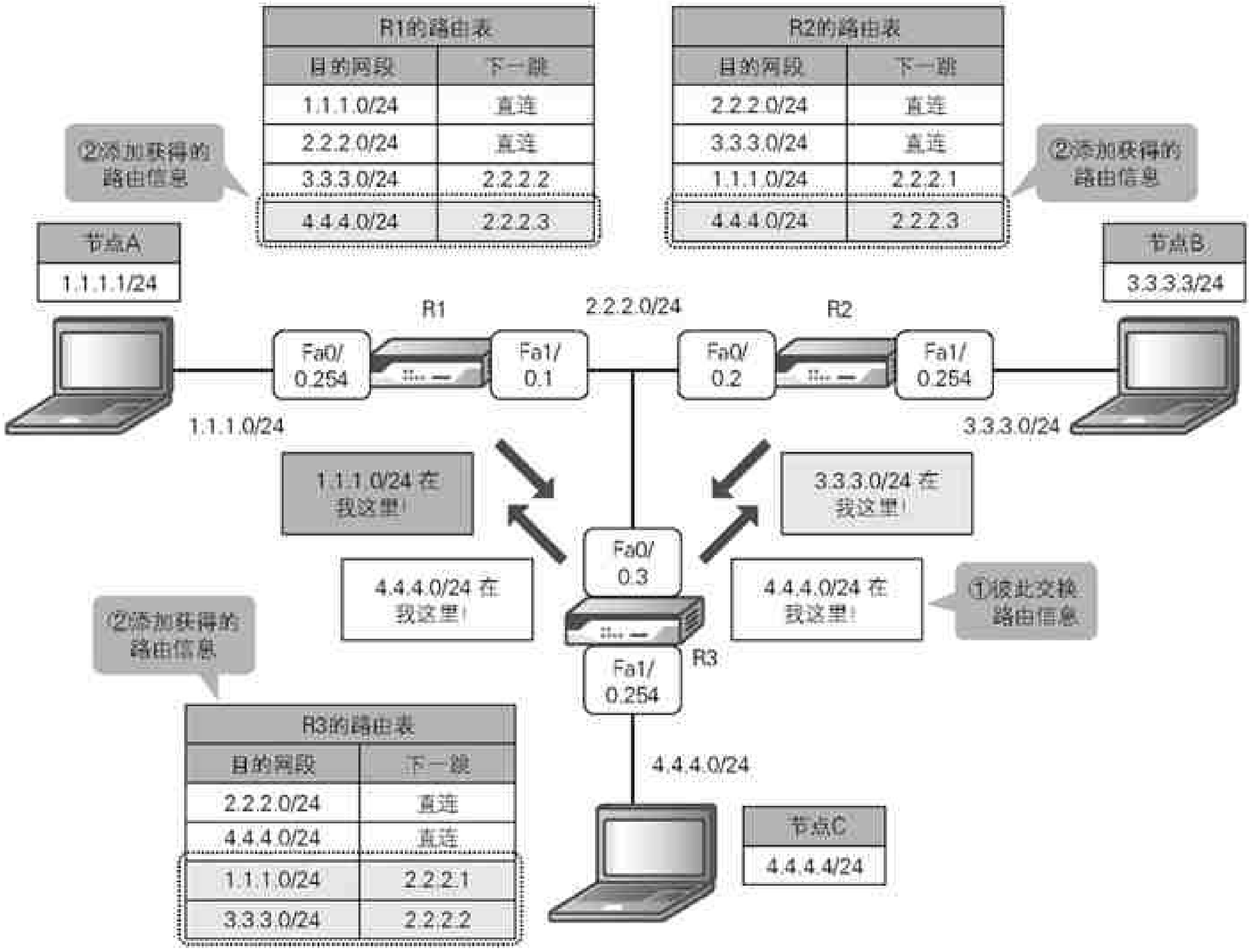


图 **2.2.28**　动态路由选择让网段的增加变得轻松简单

动态路由选择还有一个优点，那就是路由协议的耐故障能力较高。假设有多条路径可以抵达目的地址，其中某条路径发生了故障。这时候，动态路由选择能够自动更新路由表，让路由器互相通知路由表的更新情况，确保新的路径畅通无阻，这样我们就无需特意去设置迂回路径了。

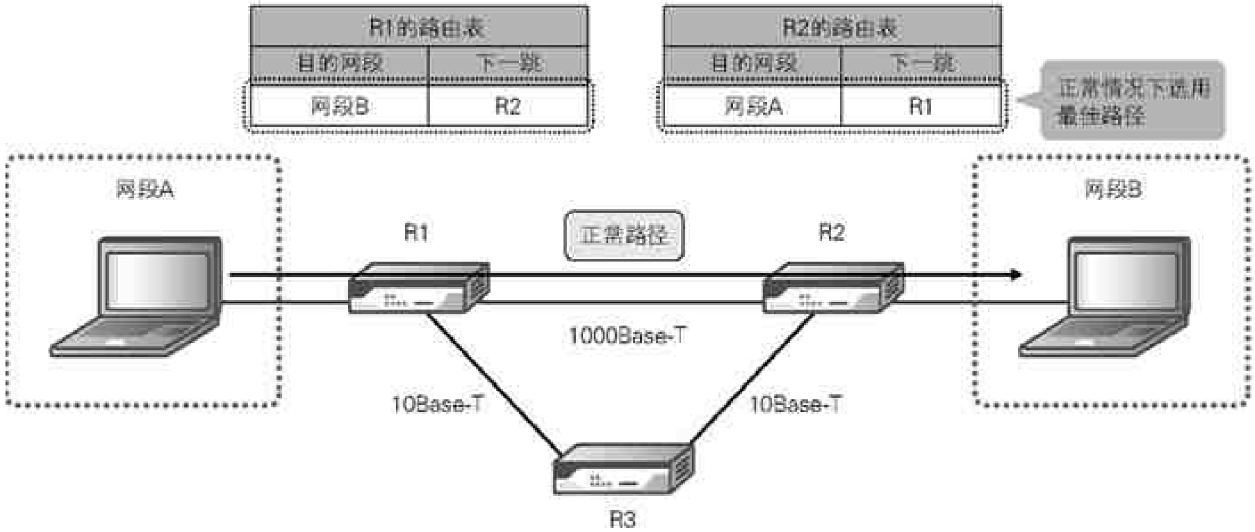


图 **2.2.29**　正常情况下选用最佳路径

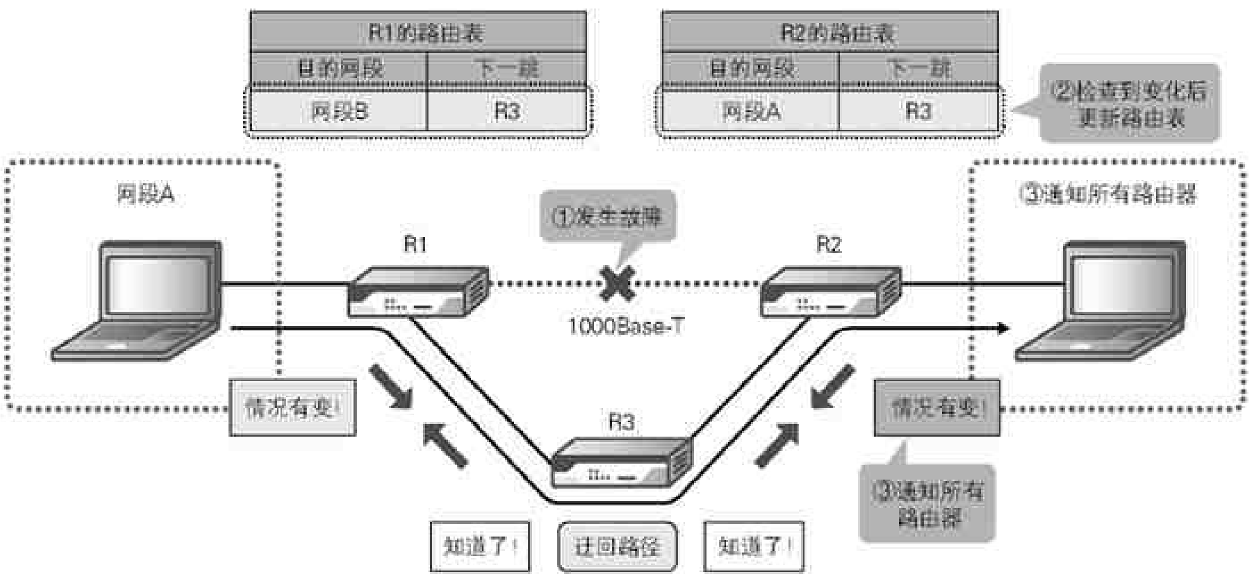


图 **2.2.30**　即使发生故障也能确保迂回路径畅通无阻

有两种路由协议

路由协议可根据控制范围分为 IGP（Interior Gateway Protocol，内部网关协议）和 EGP（Exterior Gateway Protocol，外部网关协议）两种。

将这两者区别开的概念叫作 AS（Autonomous System，自治系统），AS 是指在一个基本规则下管理的网络群组。听起来也许有点高深，但我们不必想得太复杂，将它理解为组织（ISP、企业、研究机构、网点）即

可。IGP 是控制 AS 内部的路由协议，EGP 则是控制 AS 之间的路由协议。一般说来，IGP 使用 RIPv2、OSPF 和 EIGRP，EGP 则使用 BGP。

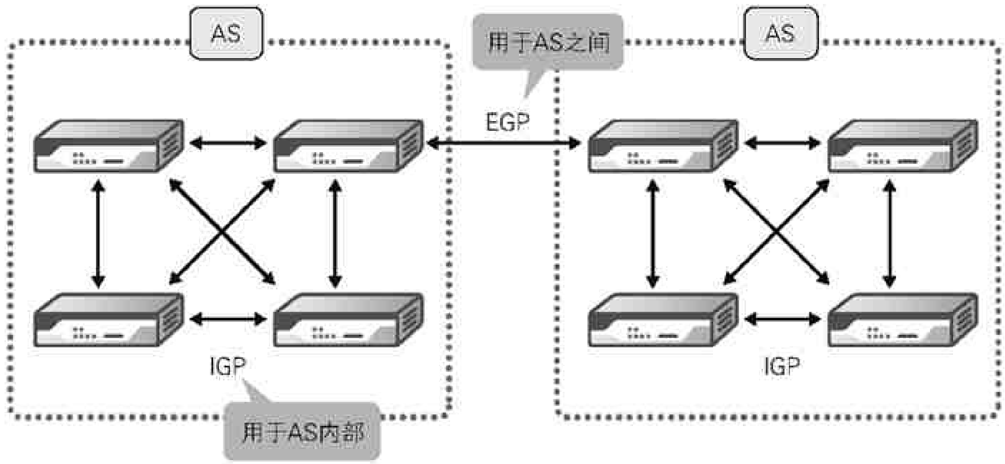


图 **2.2.31**　根据控制范围可将路由协议分成两种

IGP 的两个要点是路由算法和度量值

IGP 是用于 AS 内部的路由协议，种类繁多，不过基本上我们可以认为当前网络环境中使用的协议是 RIPv2（Routing Information Protocol version2，路由信息协议第二版）、OSPF（Open Shortest Path Fase，开放式最短路径优先）和 EIGRP（Enhanced Interior Gateway Routing Protocol，加强型内部网关路由协议）这三者当中的一个。在介绍它们时必须提到两个要点——路由算法和度量值。

- 路由算法

  路由算法表示如何建立路由表。路由算法不同将直接影响到收敛时间和适用的网络规模。 IGP 有两种路由算法，一种是距离向量型，另一种是链路状态型。

  距离向量型算法根据距离和方向计算路由，这里所说的距离是指抵达目的网段所经过的路由器台数（跳跃计数），而方向是指输出接口。经过多少台路由器才能抵达目的网段就是最佳路径的判断标准。路由器彼此交换各自的路由表信息，以此来建立路由表。

  链路状态型算法根据链路的状态计算最佳路径，路由器彼此交换各自的链路（接口）状态、带宽和 IP 地址等信息并建立起数据库，

然后根据这些信息去建立路由表。

- 度量值

  度量值表示到目的网段的距离，这里所说的距离并非物理距离，而是指逻辑距离。比如，我们和地球另一边进行通信时，度量值并不一定就很大。路由协议不同，逻辑距离的计算方法也就多种多样。

IGP 中的 RIPv2、OSPF 和 EIGRP 都要掌握

前面说过，当前网络环境中使用的路由协议无非是 RIPv2、OSPF、EIGRP 当中的某一个。笔者尚未听说除了这三者之外还有什么新的路由协议，可以说它们涵盖了 IGP 的全部。下面就结合路由算法和度量值分别说明一下这三个协议。

表 **2.2.6**　掌握好 **RIPv2**、**OSPF** 和 **EIGRP**，**IGP** 就没问题了

细分项目	**RIPv2**	**OSPF**	**EIGRP**
路由算法	距离向量型	链路状态型	距离向量型（混合型）
度量值	跳跃计数	开销	带宽＋延时
更新周期	定期更新	有变化就更新	有变化就更新
更新时使用的多播地址	224.0.0.9	224.0.0.5（都是OSPF路由器） 224.0.0.6（都是DR路由器）	224.0.0.10
适用的网络规模	小规模	中到大规模	中到大规模

- **RIPv2**

  RIPv2 是距离向量型的路由协议。这个协议最近已经很少见到了，但它依然存在于一些旧的环境，由它过渡到 OSPF 或 EIGRP 也是

常有的事。不过，现在新建网络环境时选用 RIPv2 应该是不大可能的。RIPv2 通过彼此定期交换路由信息的方式建立路由表，原理很好理解。然而随着路由表条目的增多，这种方式会过度消耗带宽，收敛时间也较长，所以并不适合大规模的网络环境。

RIPv2 采用跳跃计数作为度量值，跳跃计数表示要经过多少台路由器才能抵达目的网段，经过的路由器越多距离就越远。这个协议的原理也非常简单易懂，但是它缺点很多，例如它会不顾带宽的实际情况来判断最佳路径。也就是说，即使有的路径带宽较小，该协议也会将跳跃计数最小的路径当成最佳路径。RIP 中当然也有更古老的 RIPv1 版本，不过 v1 只能用于分级地址，现在已经完全看不到了。

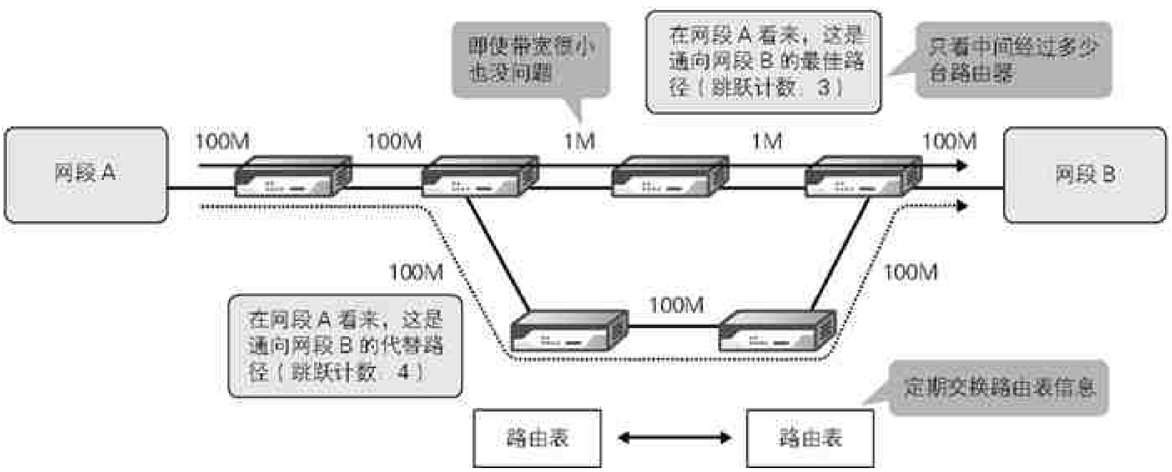


图 **2.2.32　RIP** 根据跳跃计数决定路径

- **OSPF**

OSPF 是链路状态型的路由协议。它是一种标准化了的路由协议，常常用于多厂商的网络环境。OSPF 下的路由器彼此交换各自的链路状态、带宽、IP 地址和网段等信息，建立链路状态数据库（LSDB）并通过相关数据计算最佳路由，然后建立起路由表。RIPv2 是彼此定期交换路由表信息，而 OSPF 则只在发生变化时才去更新信息。除此之外，OSPF 在正常情况下只发送一个 Hello 小数据包，用来检查对方是否处于正常运作状态，因此不会过度占用带宽。OSPF 中有一个重要的概念叫作区域。为了防止汇集了各种信息的 LSDB 因信息过多而臃肿膨胀，网络被分成了好几个区域，只有同一区域中的路由器能够共享 LSDB。

OSPF 采用开销作为度量值。开销在默认情况下使用“100/ 带宽 Mbit/s”的公式来计算。带宽越大开销就越小，每跨一个路由器就会多算一个输出接口。因此，路由器的带宽越大就越容易成为最佳路径。如果学习的路径完全相同，则选用开销相同的所有路径去传输数据包以分散负荷，这种情况叫作等价多路径。

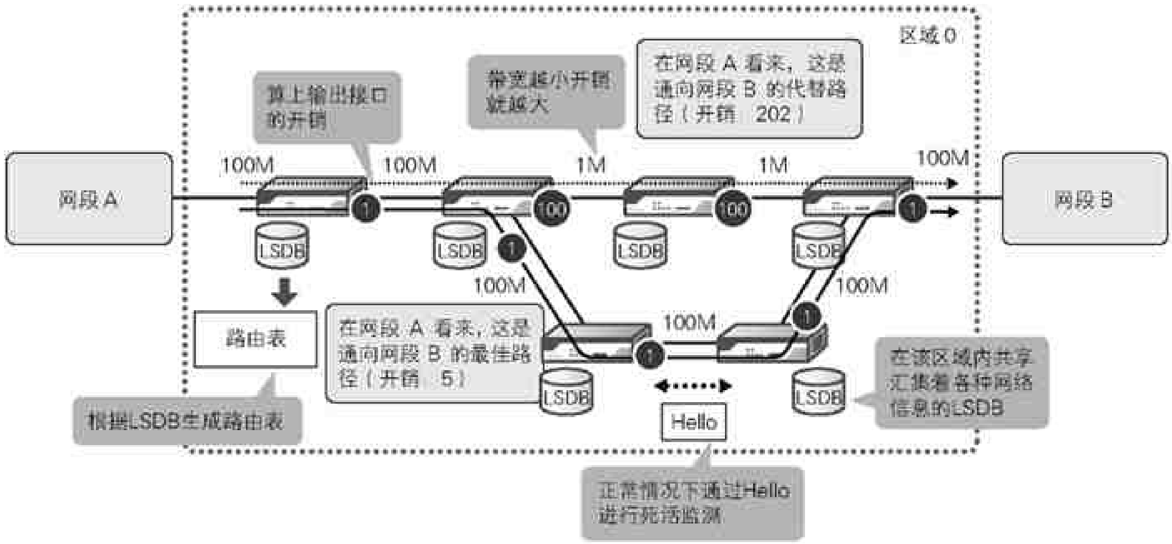


图 **2.2.33　OSPF** 根据开销决定路径

- **EIGRP**

EIGRP 是距离向量型路由协议的加强版，是思科公司特有的一个路由协议，只能在由思科路由器和 Catalyst 交换机构成的网络环境中使用。只要具备相应的环境，该协议就能发挥巨大作用。EIGRP 结合了 RIPv2 和 OSPF 各自的优点。它先是交换彼此的路由信息，各自建立拓扑表，然后从中抽取最佳路径的信息建立起路由表。这部分和 RIPv2 有点相似，不过 EIGRP 只在发生变化时才去更新路由表。此外，EIGRP 在正常情况下只会发送一个 Hello 小数据包，用来检查对方是否处于正常运作状态，这部分又和 OSPF 相似。

EIGRP 默认采用带宽和延时作为度量值。带宽用“10000/ 最小带宽（Mbit/s）”的公式计算，最小带宽为通往目的网段的路径中带宽的最小值。延时用“微秒 /10”的公式计算，每跨一个路由器就会多算一个输出接口。将这两个数字的和乘以 256 得出的数值就是 EIGRP 的度量值。EIGRP 的默认设置也是等价多路径。如果几个路径的度量值完全相同，则选中所有这些路径去传输数据包，进行负载均

衡。

EGP 中只需掌握 **BGP** 即可

EGP 是用于 AS 之间的路由协议 [12] 。前面讲过 IGP 有三种类型，EGP 却只有 BGP（Border Gateway Protocol，边界网关协议）这一种类型，所以架构服务器端时做好 BGP 即可。在介绍 BGP 时必须提到两个要点，那就是路由算法和最佳路由选择算法。

> [12] BGP 也可用于 AS 内部，用于 AS 内部的 BGP 叫作 iBGP，用于 AS 之间的 BGP 叫作 eBGP。

BGP 是路径向量型协议，根据路径和方向计算路由。这里所说的路径表示抵达目的网段所经过的 AS 数，方向则表示 BGP 对等体。经过多少 AS 才能抵达目的网段是最佳路径的判断标准之一，BGP 对等体指的是和本机交换路径信息的对象。BGP 指定了对等体后，建立一对一的 TCP 连接并在其中交换路径信息。和 BGP 对等体交换路由信息之后建立 BGP 表，根据一定的规则（最佳路由选择算法）选择最佳路径。接下来，只把最佳路径添加到路由表中去，同时传播给 BGP 对等体。BGP 和 OSPF、EIGRP 一样，仅在发生变化时才去更新路由表（通过 UPDATE 消息更新）。另外，它在正常情况下通过 KEEPALIVE 消息判断对方是否处于正常运作状态。

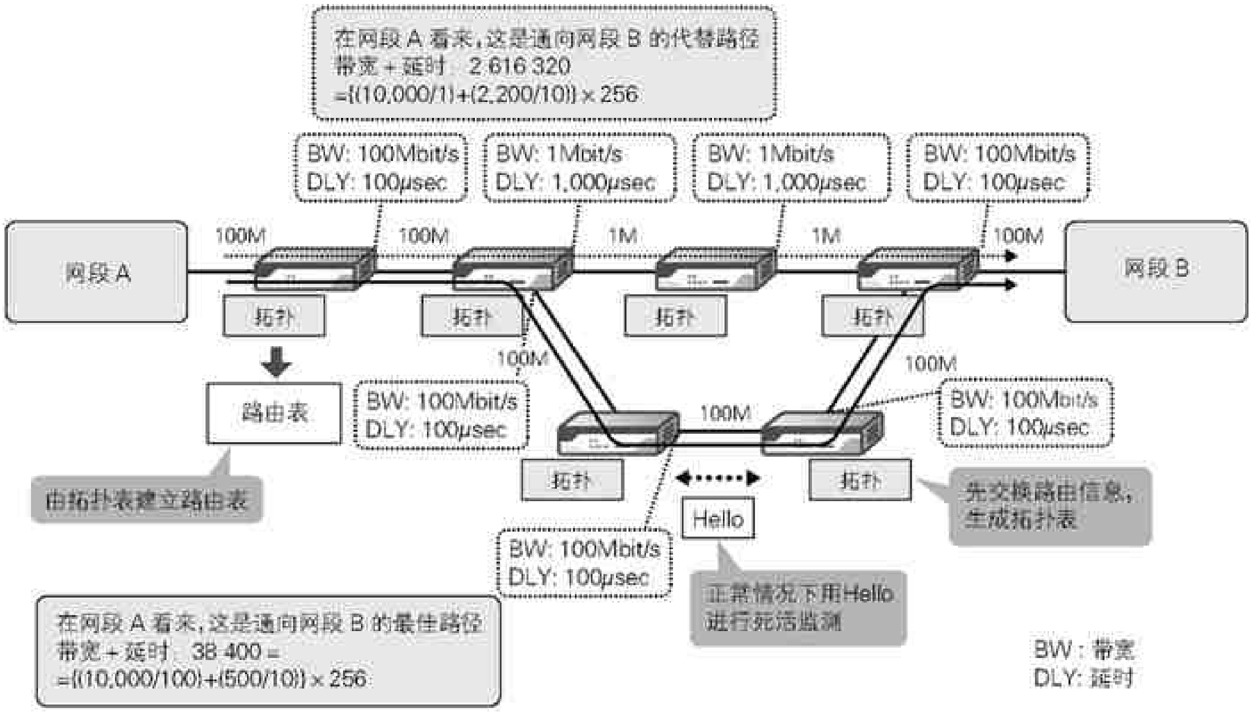


图 **2.2.34　EIGRP** 根据带宽和延时决定路径

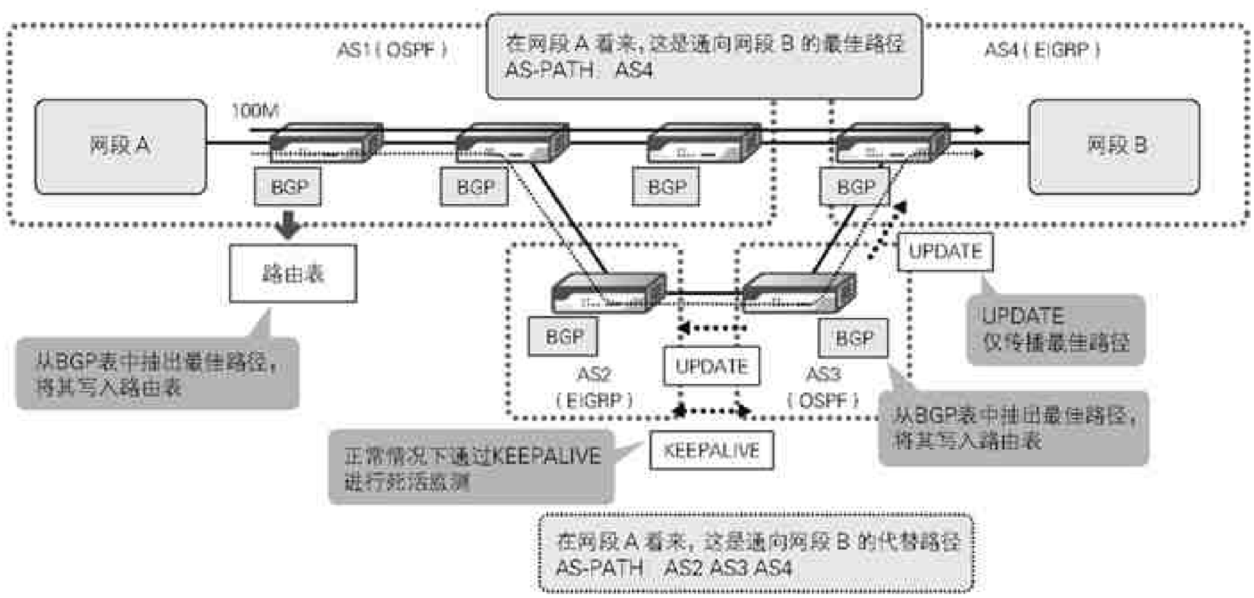


图 **2.2.35　BGP** 默认根据经过的 **AS** 数量决定路径

最佳路由选择算法是一种规则，用来判断哪条路径是最佳路径。互联网通过 BGP 将全世界所有的 AS 连接成网状，这样的范围必然会牵涉到国家、政治和金钱。为了灵活应对各种情况，BGP 中配置了多种路由控制功能。BGP 的路由控制是通过属性来实现的。UPDATE 消息中含有

NEXT_HOP、LOCAL_PREF 等各种属性，这些属性都会写入 BGP 表。然后，根据最佳路由选择算法选出最佳路径。如下图 2.2.36 所示，按从上至下的顺序进行比较和淘汰，一旦出局就不再入选。最后，将选出的最佳路径添加到路由表中去，同时传播给 BGP 对等体。

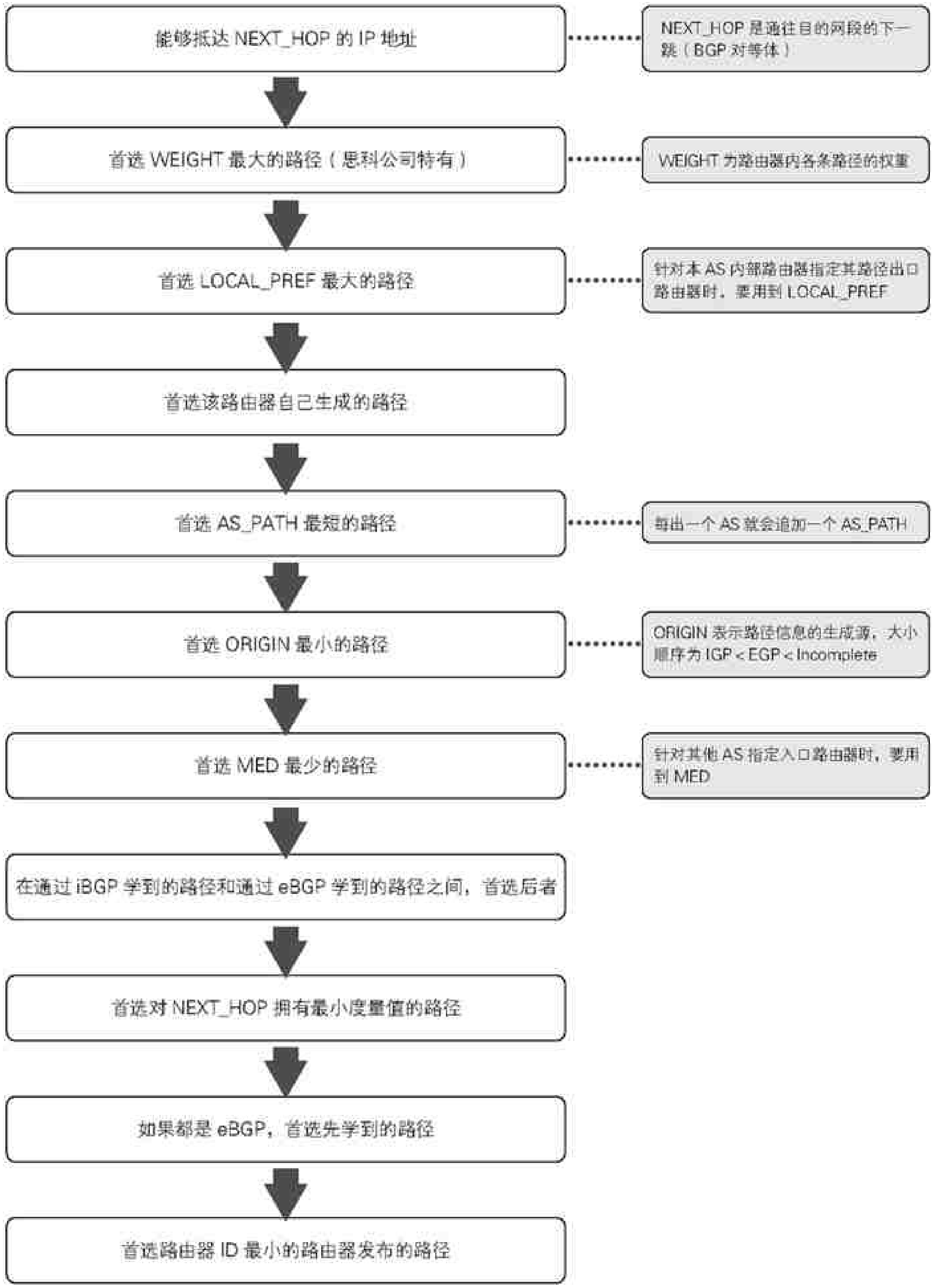


图 **2.2.36**　根据最佳路由选择算法不断进行优胜劣汰

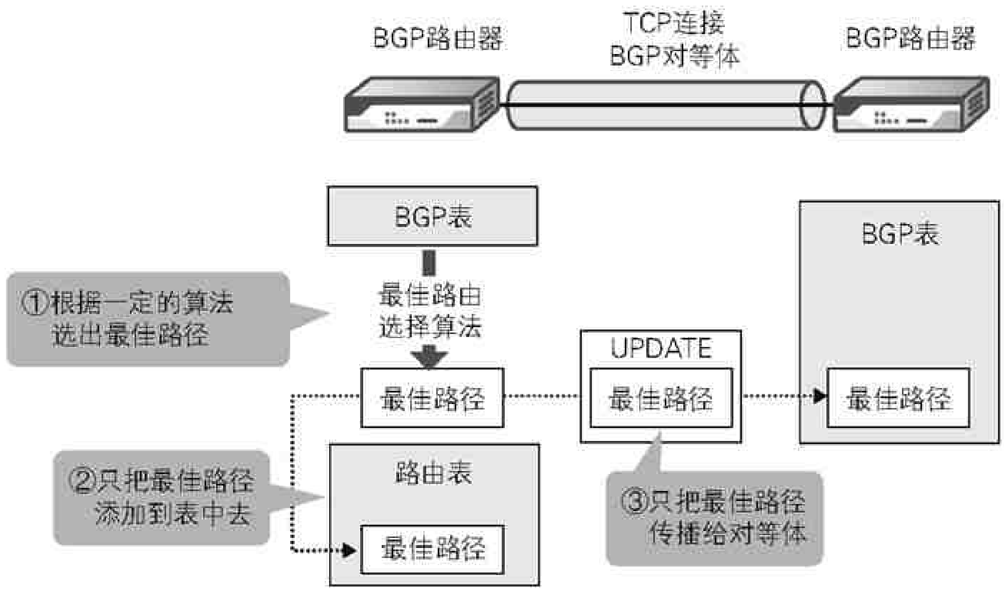


图 **2.2.37**　从 **BGP** 表中选择最佳路径

“再发布**”**让我们能够同时使用多种路由协议

前面对 RIPv2、OSPF、EIGRP 和 BGP 一一作了说明。这四种协议各自使用不同的路由算法和度量值，处理过程也各不相同，不能互相兼容。那么，架构网络时使用一个统一的路由协议，整体上也比较清晰岂不是更好？然而现实却没有那么简单。公司可能会合并或分化，现有设备可能并不支持指定的路由协议，各种特殊情况夹杂交织，常常使我们不得不同时使用多种路由协议。这时候，我们需要将它们转换一下形式，使它们能够彼此协作。这个转换作业叫作再发布，也有人称之为重分发（Redistribution），叫法和写法因人而异，但都是同一个意思。

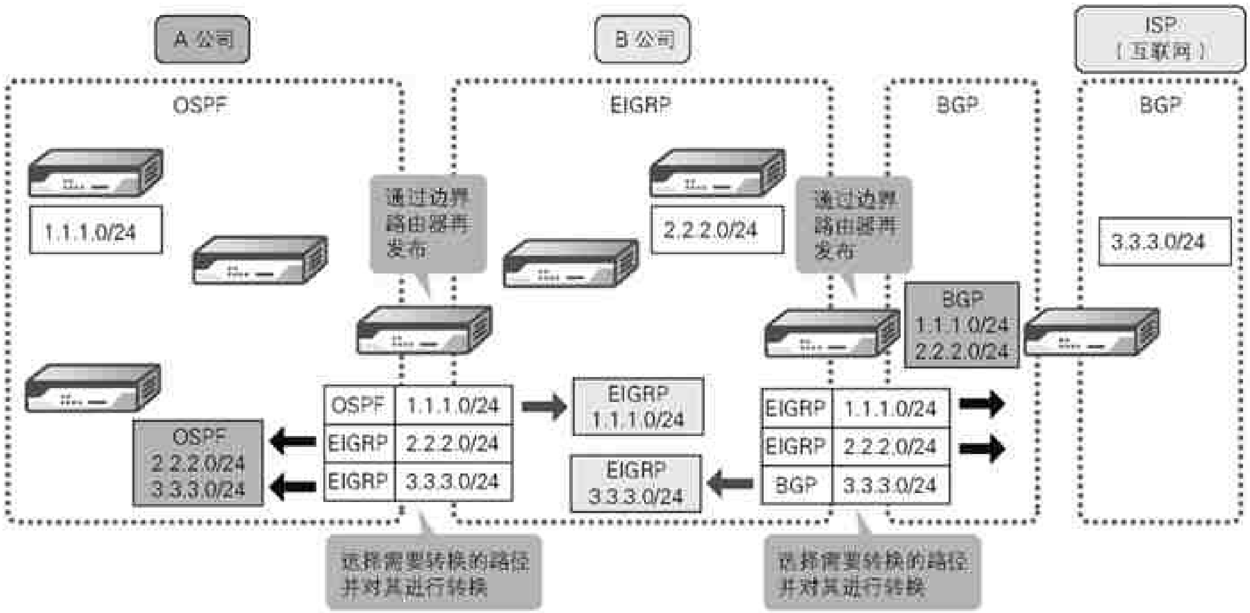


图 **2.2.38**　再发布使多种路由协议能够同时使用

再发布需要在边界路由器中设置。边界路由器是连接不同路由协议的路由器，它会选出路由表中通过路由协议学到的、需要转换的路由条目，将其转换之后传播给对等体。

2.2.2.3　整理路由表

前面着重讲解了如何建立路由表，接下来我们看看建好之后的路由表。建好之后的路由表有三个要点，分别为最长匹配原则、路由汇总和 AD 值。

优先选择更加细长的路径（最长匹配原则）

最长匹配原则是关于路由表中路由匹配的一条规则，即当有多条符合目的 IP 地址条件的路径存在时，它会选择子网掩码最长的那条路径。我们知道，路由器收到 IP 数据包之后会去查看数据包的目的 IP 地址。在最长匹配原则下，路由器对 IP 地址的查看会精确到路径条目中子网掩码的位，以此来选出最符合条件的路径，也就是子网掩码最长的那条路径，然后将数据包转发给下一跳。

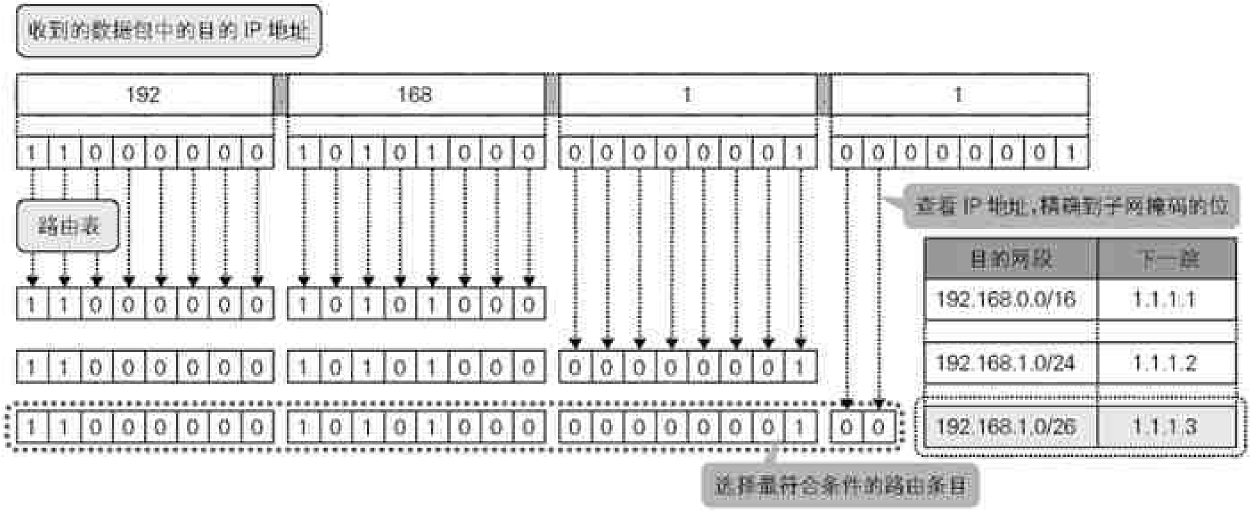


图 **2.2.39**　路由器对子网掩码的查看精确到位，以此来选出最符合条件的路径

我们以实际的网络环境为例来考查一下。假设某路由器拥有三条路径，分别是 192.168.0.0/16、192.168.1.0/24 和 192.168.1.0/26，而这台路由器收到了一个目的IP地址为 192.168.1.1 的 IP 数据包，于是每条路径都符合 192.168.1.1 这个条件，这时候就可以用到最长匹配原则。路由器会选择子网掩码最长的路径 192.168.1.0/26，将数据包转发给 1.1.1.3。

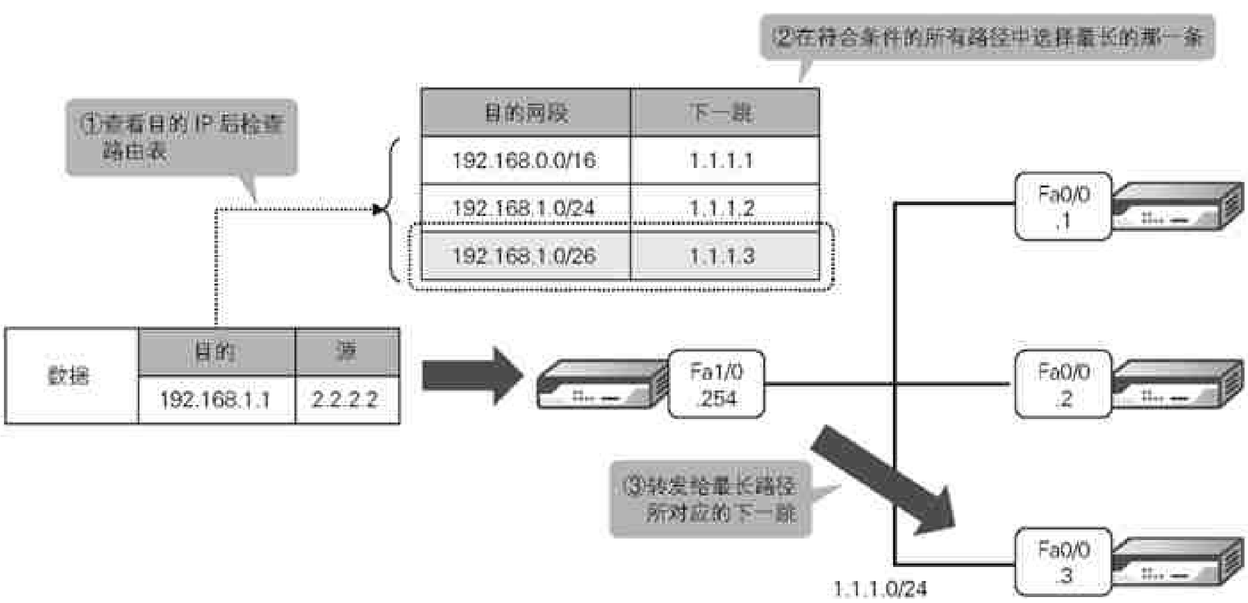


图 **2.2.40**　选择子网掩码最长的条目

“路由汇总”将多条路径汇集起来

将多条路径汇集起来的做法叫作路由汇总。路由器收到 IP 数据包后会对路由表逐条查询，随着条目的增多，作业负荷会越来越大，这是一个足以致命的弱点。为了提高使用效率，现代网络大都采用通过无分类编址划分子网的结构，而子网划分又意味着作业负荷将会随着路径的增加而增加。路由汇总就是在这样的背景下出现的，它将下一跳相同的所有路径汇集起来以减少路径的数量，进而减少路由器的作业负荷。

路由汇总的方法非常简单。它将下一跳相同的所有路径的网络地址转换成位，再将共同的位转移到子网掩码中去。举个例子，假设某台路由器拥有表 2.2.7 中所示的几条路径，在这种状态（路由汇总前）下收到 IP 数据包的话，路由器至少要对路由表查询四次。

表 **2.2.7**　路由汇总之前的路径

目的网段	下一跳
192.168.0.0/24	1.1.1.1
192.168.1.0/24	1.1.1.1
192.168.2.0/24	1.1.1.1
192.168.3.0/24	1.1.1.1

我们来对表内的路径作一下路由汇总。表 2.2.7 中的目的网段转换成位之后的结果如图 2.2.41 所示，前 22 位的位数组都是一样的，因此可以把它们汇总到 192.168.0.0/22 中去。这样，当路由器收到 IP 数据包时只需查询一次路由表就可以了。也许你会觉得不过是把四次降到了一次而已，没什么大不了的，但是积少成多、聚沙成塔，实际上有可能是几十万条路径都汇总成一条，变化之大，令人咂舌。

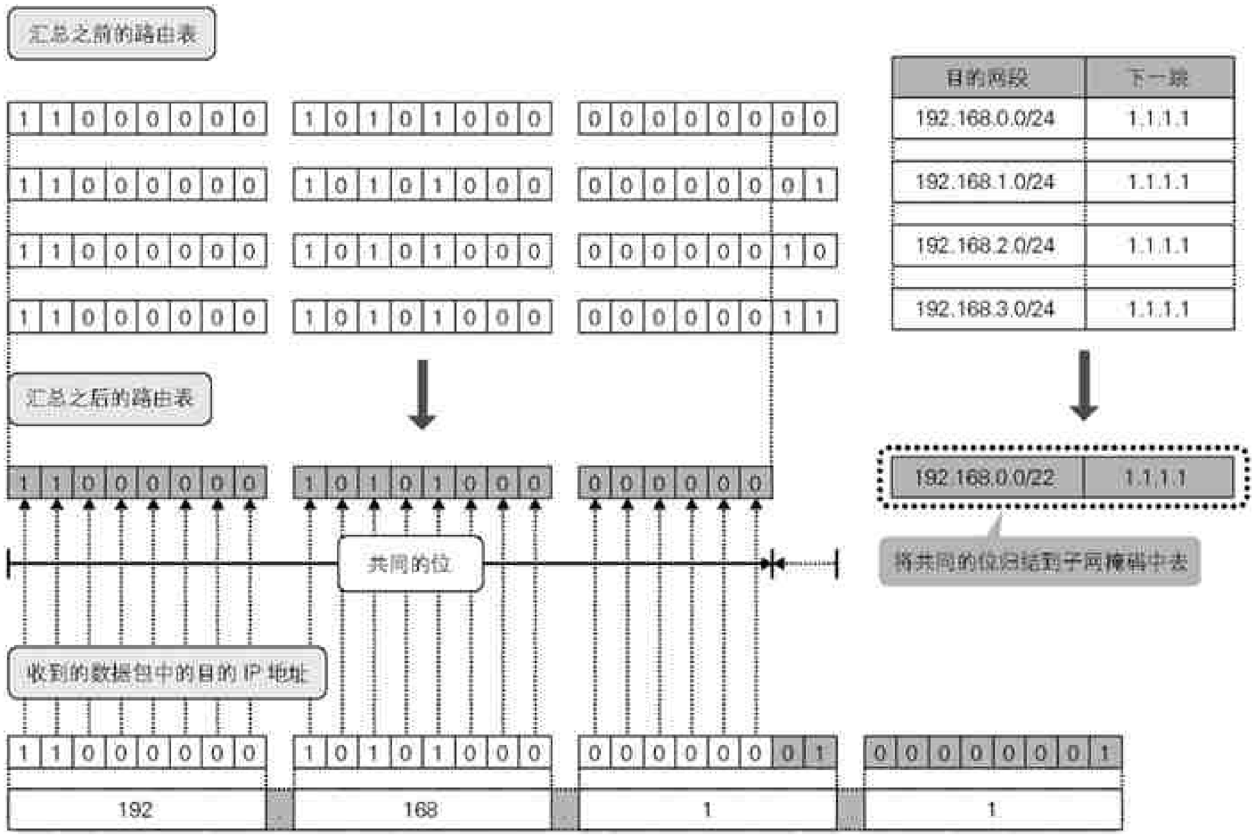


图 **2.2.41**　通过共同的位实现路由汇总

路由汇总的最高形式是默认路由。默认路由将所有路径汇总成一条，如果没有符合目的 IP 地址的条目，路由器就会将数据包发送给默认路由的下一跳，也就是默认网关。默认路由的路由条目比较特殊，是 0.0.0.0/0。

大家给 PC 设置 IP 地址时，应该也会顺便设置默认网关吧。当 PC 访问非本机所属的网段时会去查看本机中的路由表 **[13]**，由于无法在任何条目中查到目的 IP 地址，就会将数据包发送给默认网关。

> [13] PC 和服务器也都有自己的路由表。Windows 可用 route print 命令查看路由表，Linux 则是用 route 命令查看。

如果目的网段相同，那就要看 **AD** 值的比较结果了！

AD（Administrative Distance，管理距离）值类似于针对每个路由协议制定的优先等级，值越小优先的等级就越高。

通过不同路由协议或静态路由选择学到的完全相同的路径，是无法用最长匹配原则去判断和选择的，遇到这种情况时我们就要用到 AD 值。路由器会比较这些路由协议的 AD 值，从中选出 AD 值更小的，也就是优先等级更高的路径并将其写入路由表，发送时也会优先采用该路径。

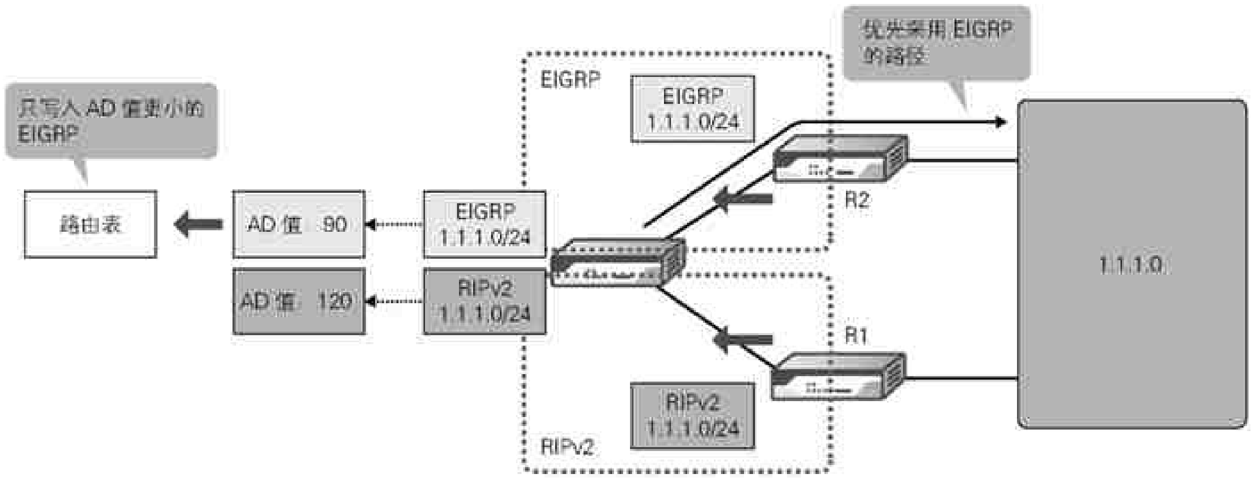


图 **2.2.42**　只有 **AD** 值更小的路径才会被写入路由表

每台机器都拥有自己的 AD 值。思科机器的 AD 默认值如下表所示，除直连协议之外都是可以修改的。AD 值还可用于防止再发布时的路由循环和浮动静态路由[14]。

[14] 这是一种路由备份的办法，只有当无法通过路由协议学习路径信息时才会使用静态路由。设置时提高静态路由的 AD 值即可实现该方法。

表 **2.2.8**　**AD** 值越小，该路径越能被优先选择

根据什么路由协议学到路径	AD 值（默认）	优先等级
直连	0	高
静态路由	1	↑
eBGP	20	
内部 EIGRP	90	
OSPF	110	
RIPv2	120	
外部 EIGRP	170	↓
iBGP	200	低

2.2.3 转换 IP 地址

转换数据包 IP 地址的技术叫作 NAT（Network Address Translation，网络地址转换）。使用 NAT 可以节省 IP 地址，使拥有同一网络地址的系统之间互通信息，解决隐藏在 IP 环境中的若干问题。

2.2.3.1 转换 IP 地址

NAT 有广义和狭义之分。广义的 NAT 指转换 IP 地址的整套技术，所指范围太广，不太好理解。这里，我们只介绍三种人们常用的 NAT，它们分别是静态 NAT、NAPT（Network Address Port Translation，网络地址端口转换）和 Twice NAT（双重 NAT）。

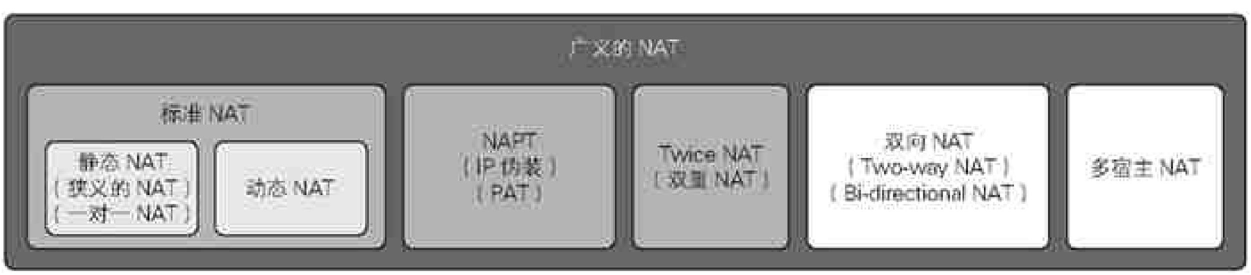


图 **2.2.43** 广义的 **NAT** 和狭义的 **NAT**

静态 **NAT** 可将 **IP** 地址一对一关联

静态 NAT 可以对内网和外网的 IP 地址进行一对一的映射和转换，又称一对一 NAT。狭义的 NAT 指的就是静态 NAT。

由内网访问外网时，静态 NAT 会转换源 IP 地址，此时是将源 IP 地址转换成相应的 IP 地址。相反，由外网访问内网时，静态 NAT 转换目的 IP 地址，此时是将目的 IP 地址转换成相应的 IP 地址。在服务器端我们有时候会遇到这样的情况，某客户端一定要以某个指定的 IP 地址出现在互联网上，或者某台服务器一定要以某个指定的 IP 地址在互联网上公开，这时候我们就要用到静态 NAT 了。

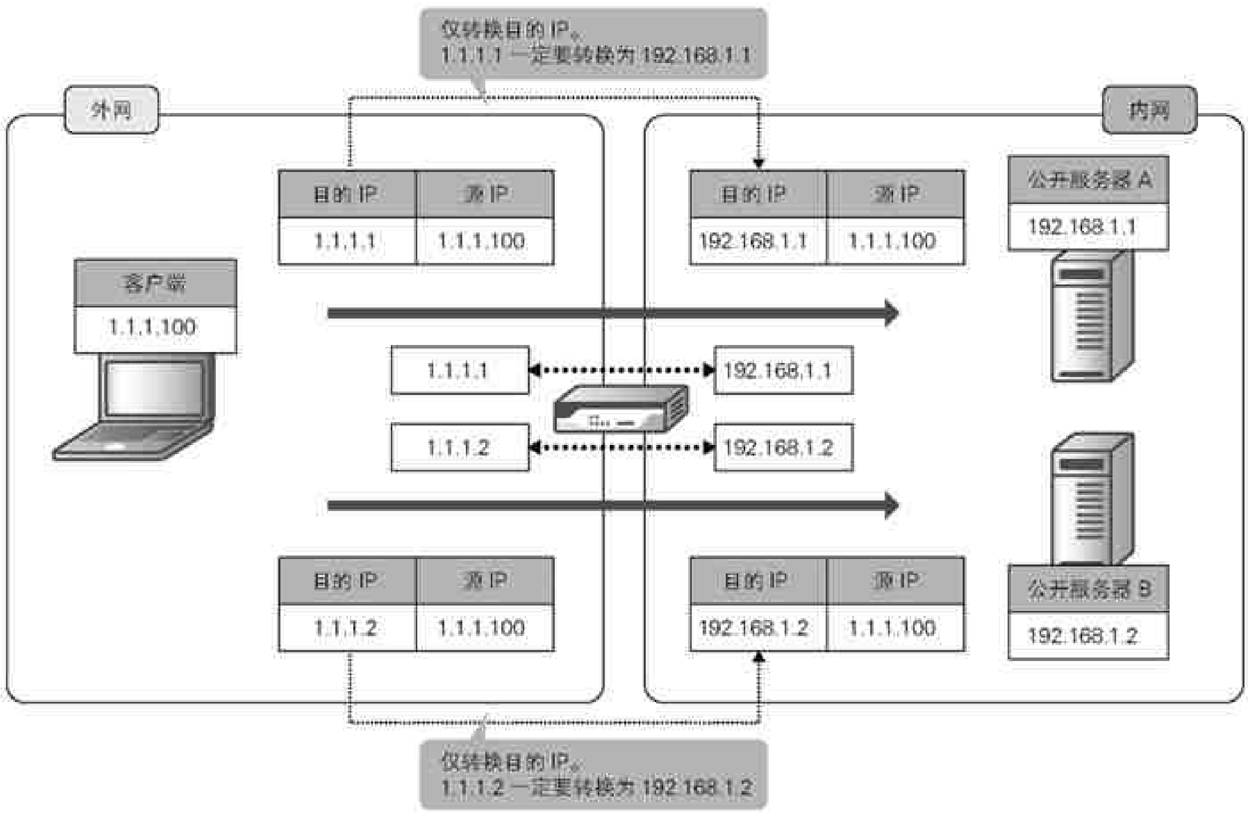


图 **2.2.44**　一对一地关联起来

NAPT 能够有效地利用 **IP** 地址

NAPT 可以对内网和外网的 IP 地址进行多对一的映射和转换，又称 IP 伪装或 PAT（Port Address Translation，端口地址转换），叫法不同，但意思都一样。

由内网访问外网时，NAPT 会将源 IP 地址和源端口号一起转换。因为

是根据客户端和端口号的对应关系（哪个客户端使用哪个端口号）来分配数据包的，所以能实现多对一的转换。

我们在家里经常使用的宽带路由就是通过 NAPT 将客户端和互联网连接到一起的。最近不仅仅是 PC，连智能手机、平板计算机和家电产品等也都开始拥有自己的 IP 地址了。如果要为这些设备一一分配具有全球唯一性的 IP 地址，地址资源很快就会告罄。所以我们要使用 NAPT，尽量节省全球 IP 地址资源。

在服务器端，由内网访问外网的通信大多都会使用 NAPT。

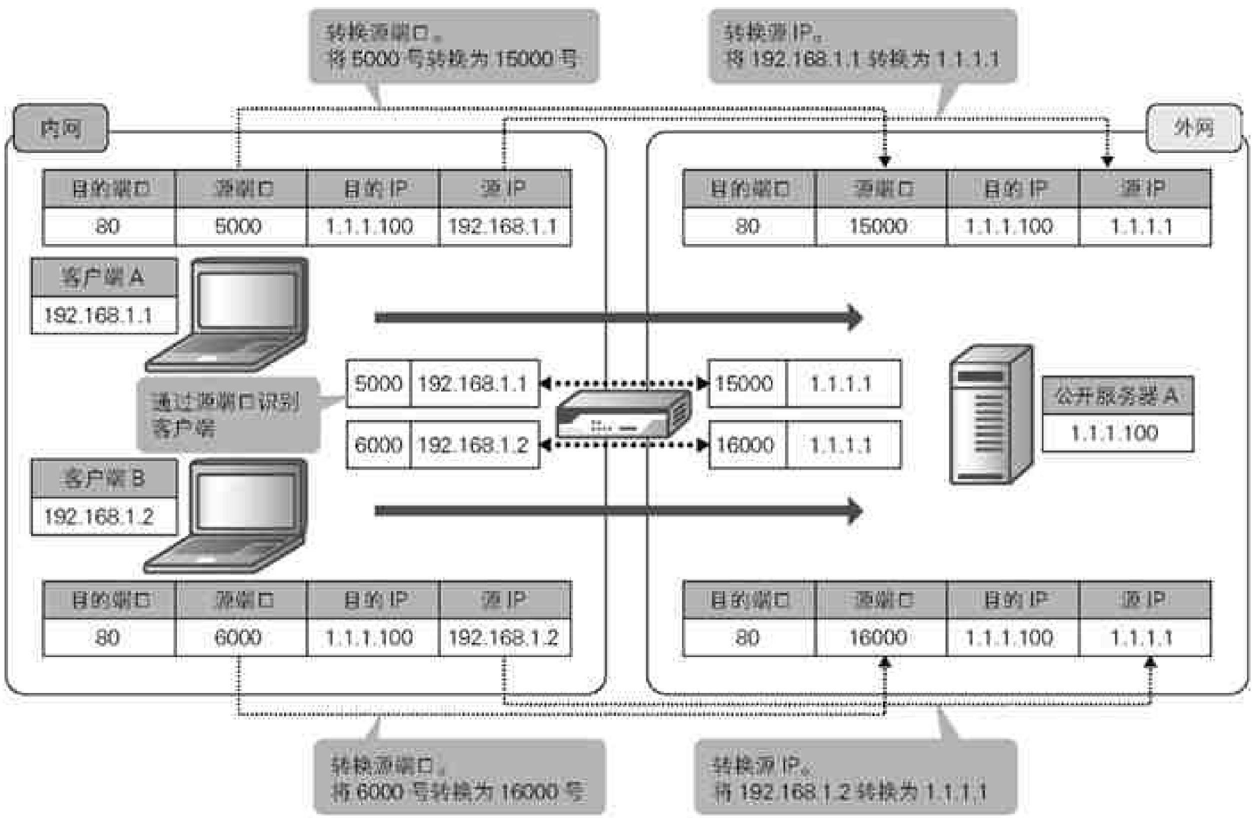


图 **2.2.45**　通过多对一的连接形式节省 **IP** 地址资源

Twice NAT 使地址范围重复的网段之间得以通信

静态 NAT 和 NAPT 只是转换源或目的地址中的其中一方而已，双重 NAT 则是将源和目的地址一起转换。当我们需要连接某个网段，而该网段的地址范围和我们本身有重复情况的时候，例如公司合并或是己方公司要和其他公司系统联网时，就要用到双重 NAT。

这里，我们来看图说话、依次讲解。假设系统 A 中有一台服务器 A，系统 B 中有一台服务器 C，服务器 C 里有着和服务器 A 相同的地址范围，而服务器 A 想要访问服务器 C（如图 2.2.46）。

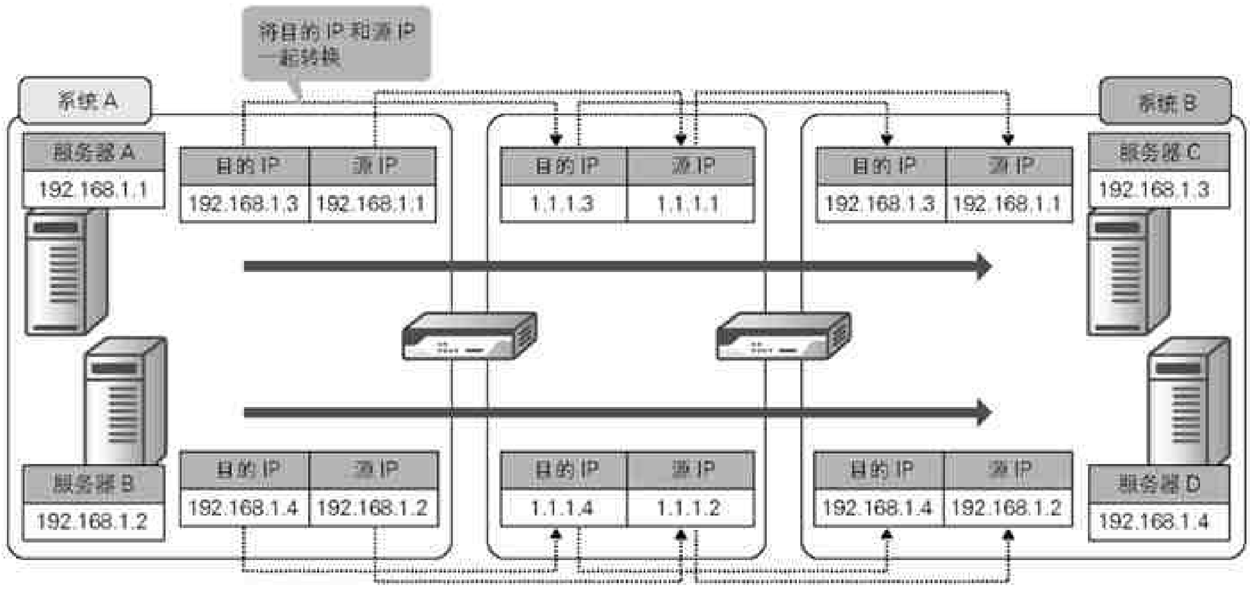


图 **2.2.46**　双重 **NAT** 使地址范围重复的网段之间得以通信

1 →　服务器 A（192.168.1.1）要访问系统 A 路由器中所设的 IP 地址（192.168.1.3），该 IP 地址用于 NAT。

2 →　系统 A 的路由器将源 IP 地址转换为 1.1.1.1，将目的 IP 地址转换为 1.1.1.3，然后将信息发至系统 B 的路由器。

3 →　系统 B 的路由器将源 IP 地址转换为 192.168.1.1，将目的 IP 地址转换为 192.168.1.3，然后将信息发至服务器 C。

NAT 表和代理 **ARP** 是支持 **NAT** 的两大要素

上面介绍了各种类型的 NAT。支持 NAT 的两大要素是 NAT 表和代理 ARP，下面我们来分别了解一下。

我们先来看 NAT 表。NAT 表中记录着转换之前和转换之后的 IP 地址或端口号。它的原理十分简单，收到需要进行转换处理的数据包后将数据包的 IP 地址或端口号转换一下，仅此而已。如果是 NAPT，那么 NAT 表中就还需要记录端口号信息。NAT 表中记录的信息会发生动态

变化，因此，当表中的信息不再使用并到达一定的时限之后，NAPT 就会将其删掉。

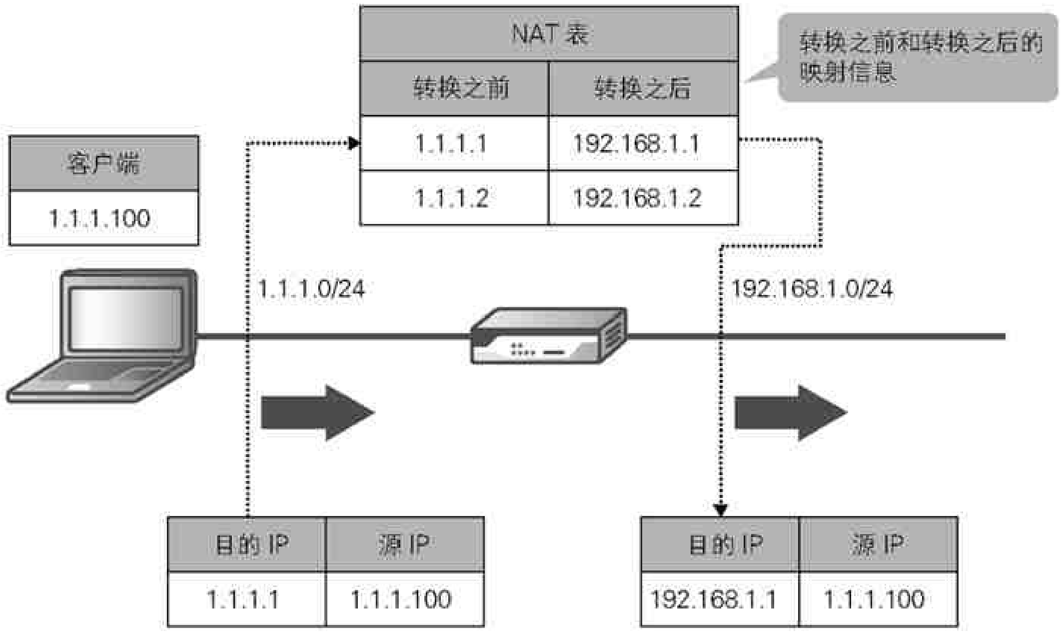


图 **2.2.47**　**NAT** 表管理着映射信息

接下来我们来看代理 ARP。用于 NAT 的 IP 地址和路由器的接口 IP 地址是不一样的，因此，对于前者发出的 ARP 请求，路由器必须作为代理给出回复。路由器以代理身份作出回复并接收前者发来的 IP 数据包，然后根据 NAT 表中的映射信息进行转换处理。

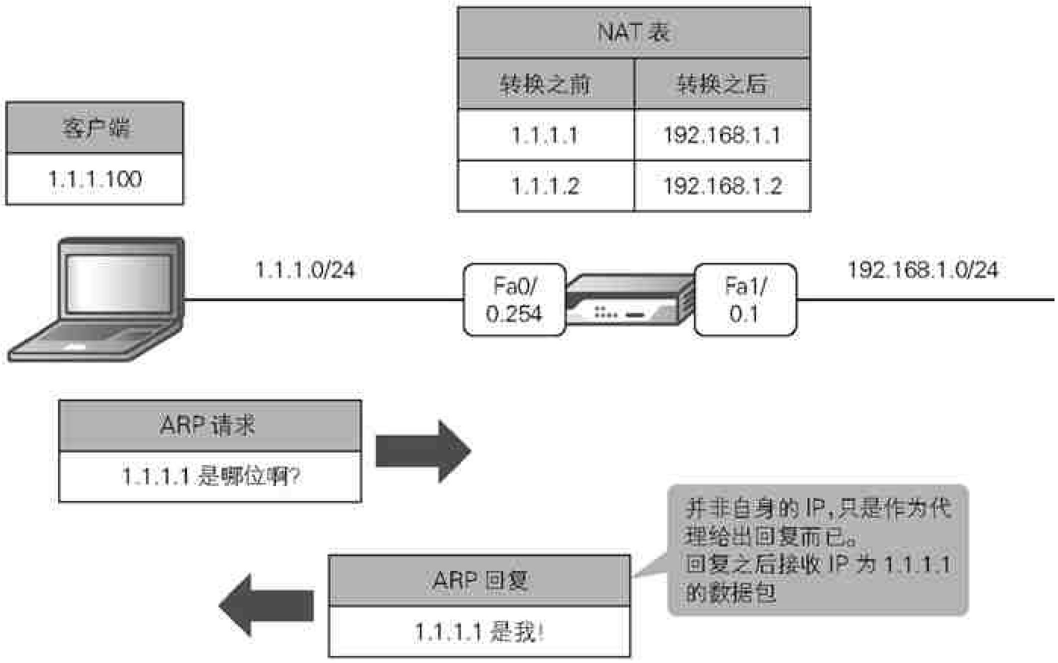


图 **2.2.48**　用于 **NAT** 的 **IP** 地址发来 **ARP** 请求时，路由器会给出代理回复

2.2.3.2　私网 **IP** 地址

可供分配的 IPv4 地址大约有 43 亿个（232）。在互联网初创期，人们相信有这么多的话肯定足够了，然而随着互联网爆发性地普及和发展，地址数量骤然狂减，人们担心这样下去的话不久地址就会告罄，于是 NAPT 应运而生了，它能够更加有效地利用现有的 IP 地址，进而减缓了地址数量的剧减。

使用 NAPT 时，不管内网中有多少台终端机，最终出现在互联网上的 IP 地址始终只有一个，只要保证这一个 IP 地址在互联网上是独一无二的就可以了。这个 IP 地址叫作全球 IP 地址，由 ICANN 及其下属机构负责其管理和分配。

那么，内网的 IP 地址应如何分配呢？是不是随便分配一下就可以了呢？回答是否定的。如果内网的终端机刚好被设置成了全球 IP 地址，那么，当它遇到早已存在于互联网并且拥有真正全球 IP 地址的服务器时，就会连接不上。为了避免这类情况，人们准备了一些专供内网分配的 IP 地址，这些 IP 地址叫作私网 IP 地址。私网 IP 地址并不会出现在互联网上，因此，即使 A 公司和 B 公司的私网 IP 地址重复了也没有任

何问题。而且，私网 IP 地址在互联网上是并不存在的。

私网 IP 地址分成三类，如下表所示。我们应根据实际的网络规模选择合适的地址范围，进行合理的子网划分。

表 **2.2.9**　私网 **IP** 地址共分三类

类别	起始 **IP** 地址	结束 **IP** 地址	子网掩码	最大可分配的节点数量
A 类	10.0.0.0	10.255.255.255	255.0.0.0	16 777 214（$=2^{24}-2$）
B 类	172.16.0.0	172.31.255.255	255.240.0.0	1 048 574（$=2^{20}-2$）
C 类	192.168.0.0	192.168.255.255	255.255.0.0	65 534（$=2^{16}-2$）

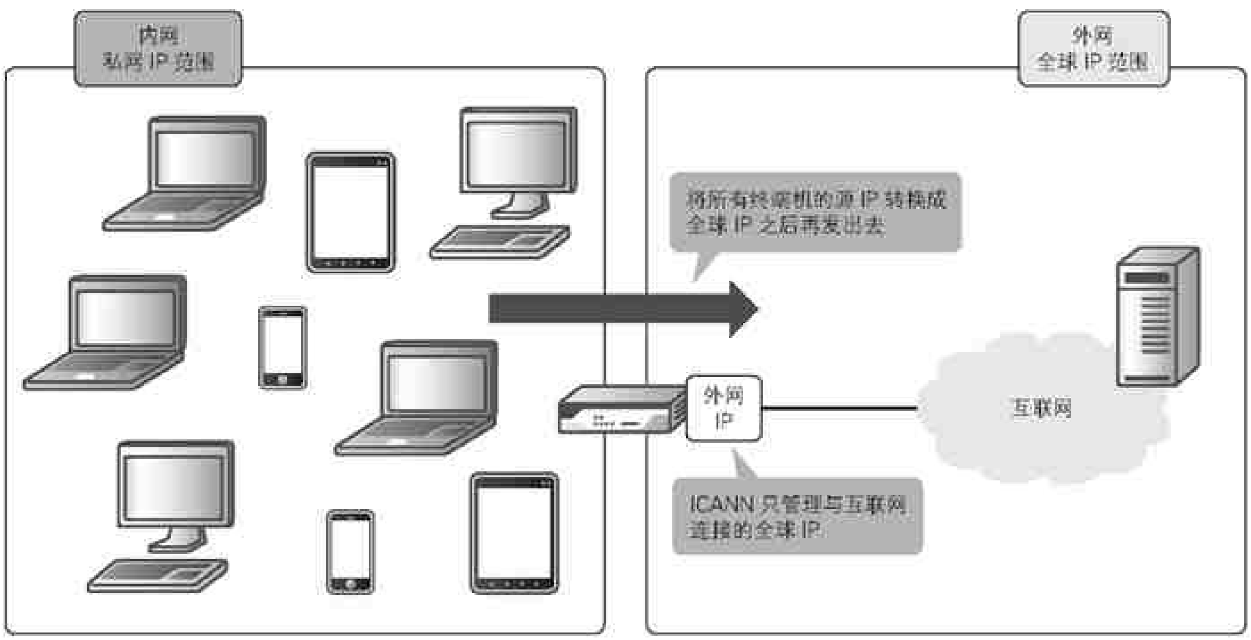


图 **2.2.49**　对内网分配私网 **IP** 地址

2.2.4　自动设置 **IP** 地址的 **DHCP**

下面介绍几个和网络层相关的协议，首先我们来看 DHCP（Dynamic Host Configuration Protocol，动态主机配置协议）。DHCP 是将 IP 地址和默认网关等网络设置分配给节点的协议。该协议本来是在应用层发挥作用的，但是由于它涉及 IP 地址的设置，所以本书将它作为网络层的技术来讲解。DHCP 不仅能使 IP 地址的管理更加轻松，还能让资源已经非常紧缺的 IP 地址发挥出最大的效用。

2.2.4.1　**DHCP** 的消息部分中包含着诸多的信息

DHCP 在被 UDP 封装的 DHCP 消息部分中放入了大量的设置信息。这个消息部分的内容稍微有些复杂，其中最重要的有三个因素，分别为“分配给客户端的 IP”“DHCP 服务器 IP”“选项”。

“分配给客户端的 IP”中写入了服务器端分配给终端的、有待设置的 IP 地址，“DHCP 服务器 IP”中写入了 DHCP 服务器的 IP 地址[15]，“选项”中写入了消息类型（Discover/Offer/Request/Ack）、子网掩码、默认网关、DNS 服务器的 IP 地址等和网络设置相关的各种信息。

[15] 如果使用 DHCP 中继代理功能，则 DHCP 服务器的 IP 地址为 0.0.0.0，这时候 Relay Agent IP Address 部分中将写入 DHCP 中继代理组件的 IP 地址。关于 DHCP 中继代理将在 2.2.4.3 节中详细说明。

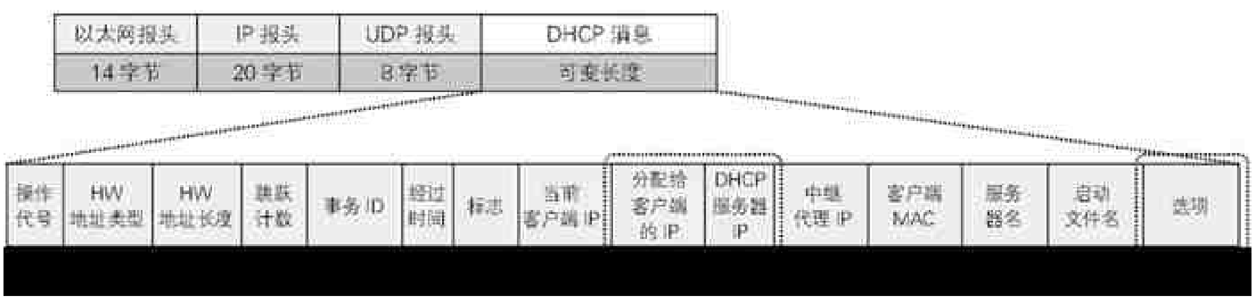


图 **2.2.50**　**DHCP** 的报文格式

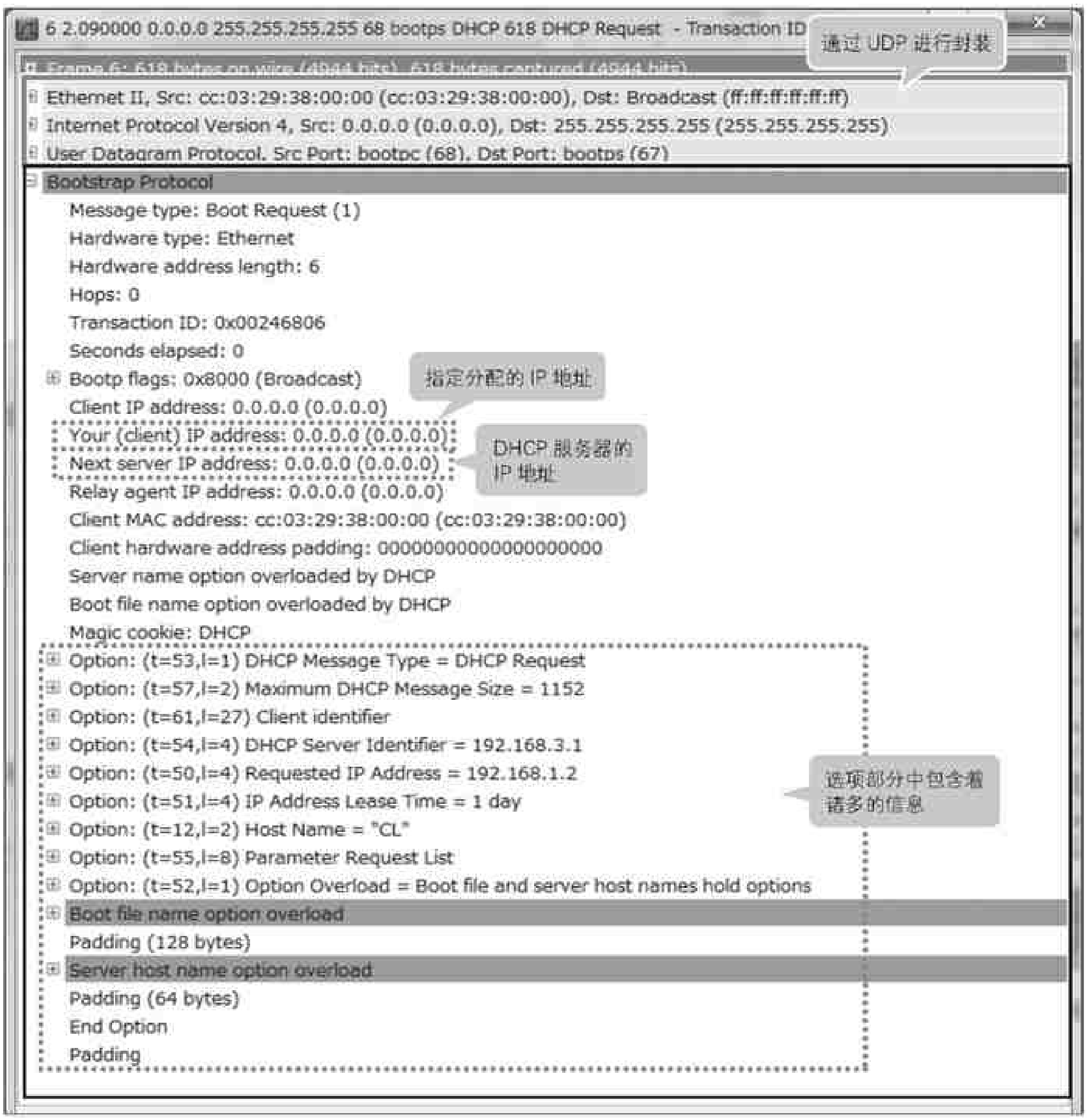


图 **2.2.51**　用 **Wireshark** 分析 **DHCP** 报文的画面

2.2.4.2　DHCP 的原理非常简单

DHCP 的原理非常简单，很好理解。请想象一下这个场景：你大声地（广播）要求“请分一个 IP 地址给我”，于是 DHCP 服务器回答“这个给你”。DHCP 就是这样一个过程。由于是在 IP 地址尚未设置的情况下通信，所以这里 DHCP 必须通过广播来交换信息。

1 →　客户端发出一个叫作 DHCP Discover 的报文用来寻找 DHCP 服务

器。

2 → DHCP 服务器收到 DHCP Discover 之后，返回一个叫作 DHCP Offer 的报文用来提议分配哪个 IP 地址。最近有些 DHCP 服务器为了确认提议的 IP 地址尚未用在别处，会在返回 Offer 之前发出 ICMP 报文，这个部分因 DHCP 服务器规格不同而有所差异。

3 → DHCP 客户端收到 DHCP Offer 之后返回一个叫作 DHCP Request 的报文，请求服务器"给我这个 IP 地址"。如果是从多台 DHCP 服务器收到多个 Offer，就会针对最早收到的那个返回 Request。

4 → DHCP 服务器收到 DHCP Request 之后，返回一个叫作 DHCP Ack 的报文，将该 IP 地址交给客户端。

5 → DHCP 客户端收到 DHCP Ack 之后，将之前收到的 DHCP Offer 中提议的 IP 地址设成自己的 IP 地址，并开始通过该地址通信。收到的 IP 地址中设有租约时间，一旦超过租约时间，客户端就会发出一个叫作 DHCP Release 的报文，解除该 IP 地址并将其返还给 DHCP 服务器。

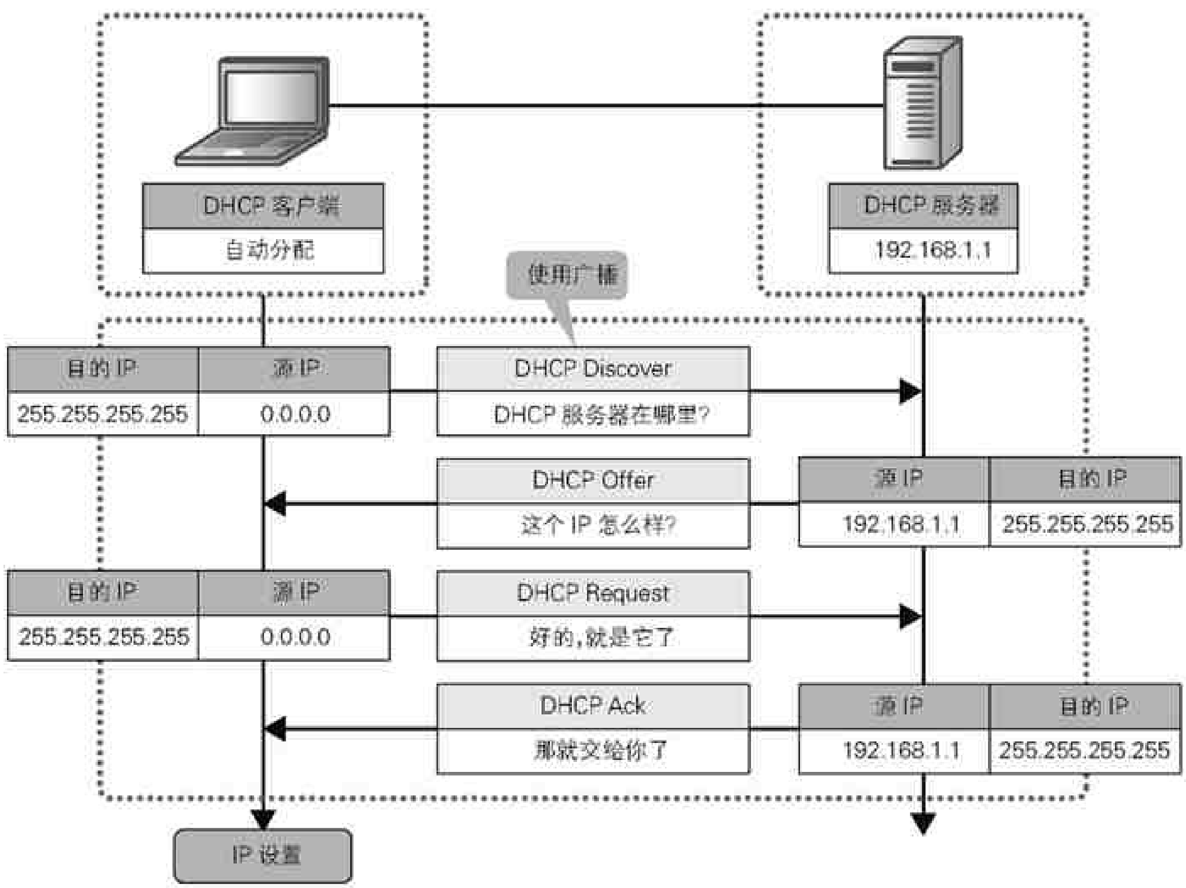


图 **2.2.52　DHCP** 通过广播来交换信息

2.2.4.3　对 **DHCP** 报文作中继处理

DHCP 通过广播交换信息，因此，原则上客户端和服务器必须在同一 VLAN 中才行。然而，在拥有大量 VLAN 的网络环境中为每个 VLAN 都安排服务器是不太可能的，这时候就要用到 DHCP 中继代理功能。DHCP 中继代理功能可以将通过广播收到的 DHCP 报文转换为单播的形式，广播变成单播之后，即使 DHCP 服务器在不同的 VLAN 中也能够对 IP 地址进行分配。而且，VLAN 再多也可以只通过一台 DHCP 服务器去分配 IP 地址。

我们应在 DHCP 客户端所在的第一跳（即默认网关的路由器）或 L3 交换机那里开启 DHCP 中继代理功能。路由器（L3 交换机）收到 DHCP 报文后，会将源 IP 地址转换为本机的 IP 地址，将目的 IP 地址转换为 DHCP 服务器的 IP 地址，进行路由。

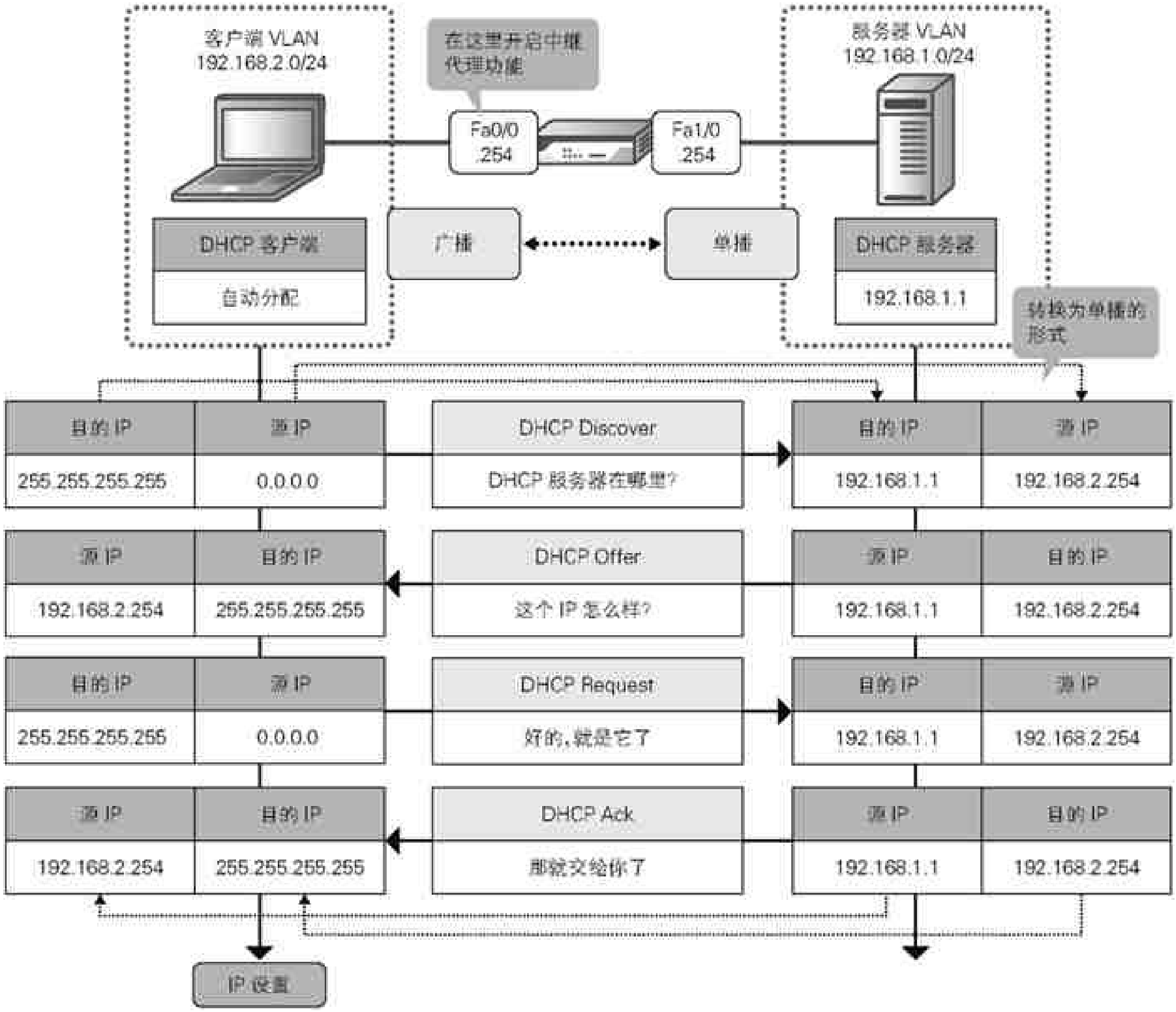


图 **2.2.53**　通过 **DHCP** 中继代理功能将 **DHCP** 报文发送到其他 **VLAN** 中去

2.2.5　用于故障排除的 ICMP

ICMP（Internet Control Message Protocol，互联网控制报文协议）是用于确认网络层通信状况的协议。当 IP 方面发生故障或是数据包未能正常送达目的网段时，ICMP 会将这些情况传达给信息发送方。从事基础架构工作的读者一定听说过 ping 这个词吧，ping 指的就是为发送 ICMP 报文使用的一种网络诊断程序。

2.2.5.1　ICMP 的关键在于类型和代码

ICMP 既不是 TCP 也不是 UDP，只代表着 ICMP 本身。它是在 IP 报头中加上了 ICMP 数据的 IP 数据包，格式并不复杂，其中有两个关键的要素，分别是类型和代码。类型表示 ICMP 报文的类 型，发送 ping（请求）和返回 ping（回复）时会使用不同的报文类型。代码分好几种类型，用于向源地址提供更为详尽的信息。例如，当返回 Destination Unreachable（类型 3）时，就是通过代码告诉源地址为什么数据未能送达目的网段。

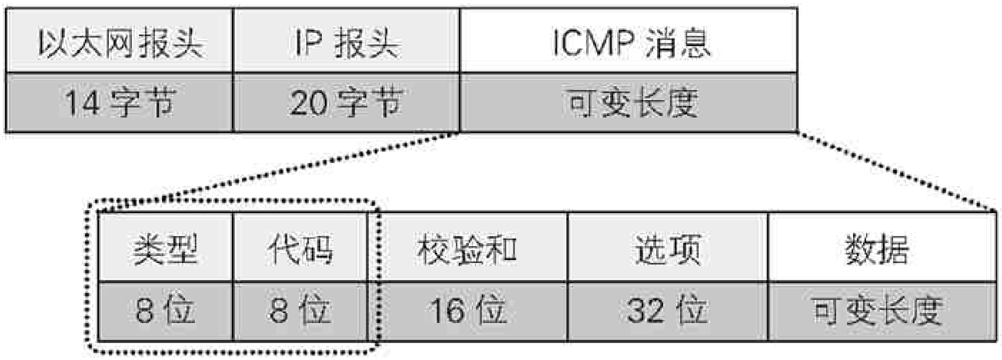


图 **2.2.54　ICMP** 的报文格式



图 **2.2.55**　用 **Wireshark** 分析 **ICMP** 报文的画面

2.2.5.2　常见的类型和代码有四种组合

类型和代码控制着 ICMP，通过这两者的组合，我们可以大致了解 IP 层面发生了怎样的情况。二者的组合种类繁多，一一说明没有多大意义，本书仅介绍几种有代表性的组合，也就是典型的几种通信类型。

通信成功则显示**“Reply from** 目的 **IP** 地址**”**

我们来看看 IP 层面上通信成功的通信类型。首先，源节点向目的节点发送 ICMP，这时的 ICMP 类型为 Echo Request，类型 8/ 代码 0。无论在什么情况下，首次发送信息的 ICMP 类型都是 Echo Request，如果通信成功，则目的节点返回类型 0/ 代码 0 的 Echo Reply。收到 Echo Reply 后，源节点上会显示出“Reply from（目的 IP 地址）”的信息。

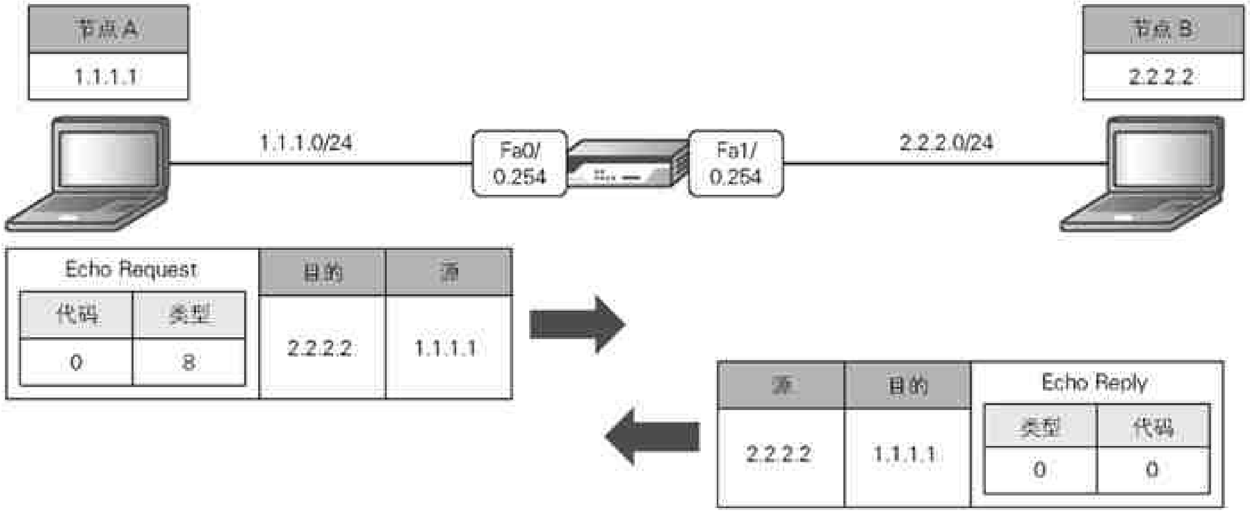


图 **2.2.56**　返回 **Echo Reply** 表示通信成功

没有回复则显示 **Request Timeout**

下面我们再来看看通信失败时的通信类型。首先，源节点向目的节点发送 Echo Request，这时的 ICMP 类型为 Echo Request。假如这个 Echo Request 未能送达目的节点会怎样呢？答案是不会返回 Echo Reply，而且不止是不会返回 Echo Reply，而是什么都不会返回。等超过时限之后，源节点会在终端上显示出 Request Timeout 的信息。时限因 OS 而异，Windows 为 4 秒，Linux 则根本没有时限的设置。

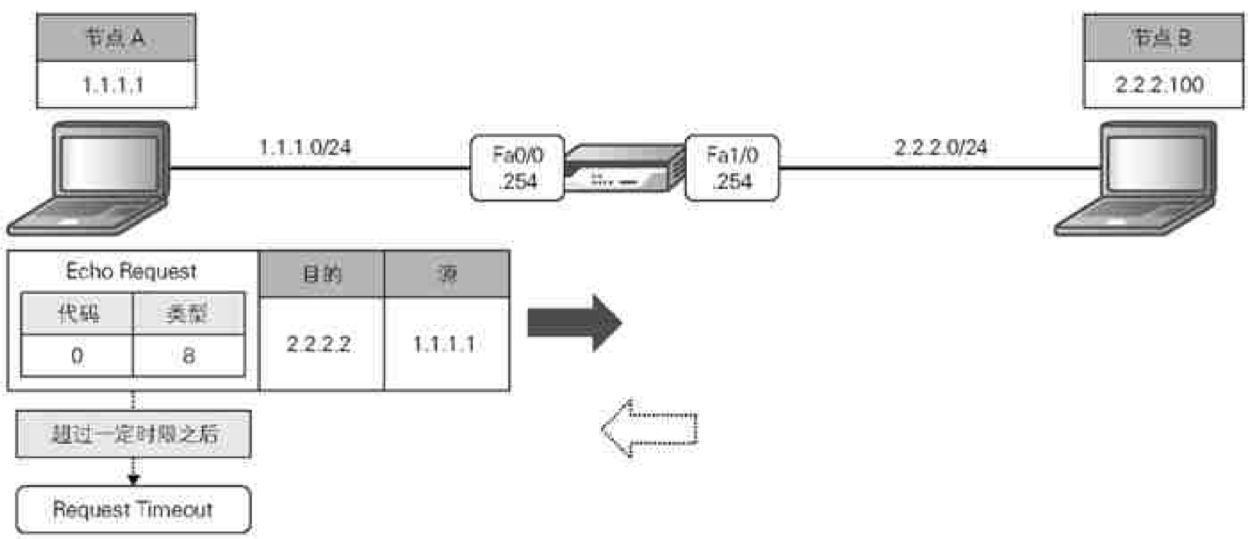


图 **2.2.57**　没有回复则显示 **Request Timeout**

查询不到目的地址则显示 **Destination Unreachable**

Destination Unreachable 是类型为 3 的 ICMP 报文。代码中显示的值表示数据未能送达的原因。

Destination Unreachable 也是一种表示通信失败的通信类型。首先，源节点向目的节点发送 Echo Request，位于通信途中的路由器会收到这个 Echo Request，假如该路由器的路由表中查询不到通往目的节点的路径，那么数据包就会被丢弃。这时路由器会返回一个 Destination Unreachable（类型 3）/Host Unreachable（代码 1）的 ICMP 报文，表示查询不到通往该目的 IP 地址的路径。

类型 3 的代码部分可能会写入好几种值。源节点查看这里就能大致明白为什么数据未能送达。

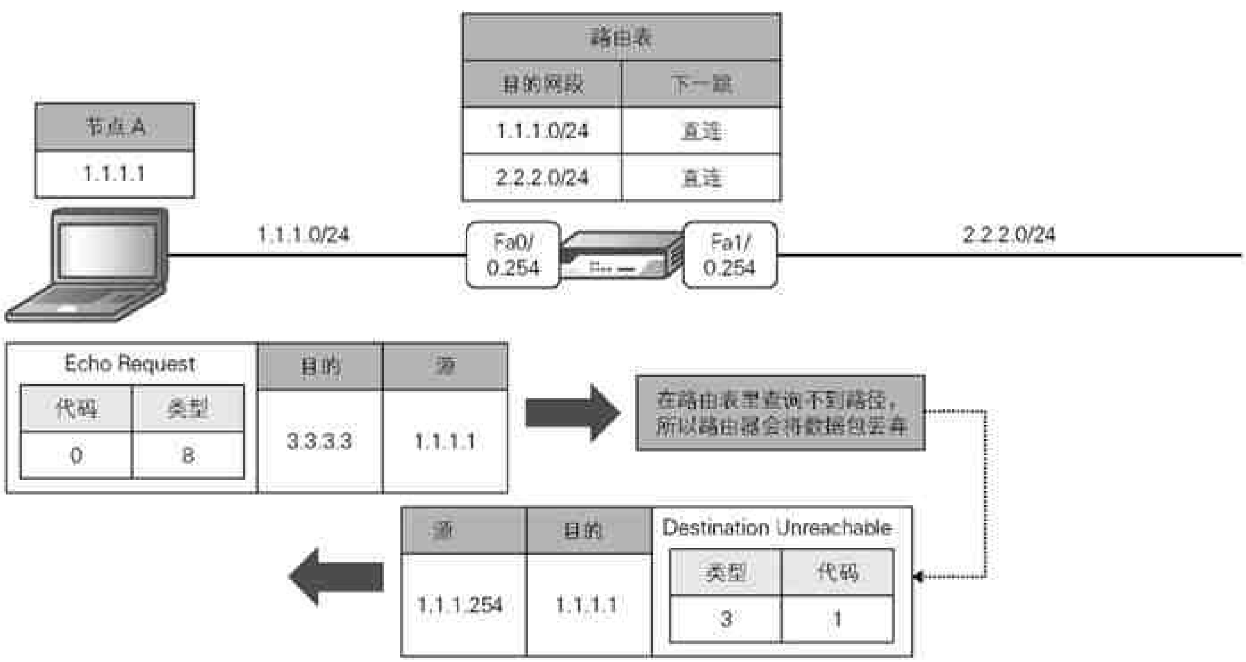


图 **2.2.58　Destination Unreachable**（类型 **3**）表示路径不存在

Redirect 将其他网关的 **IP** 地址告诉节点

Redirect 是类型为 5 的 ICMP 报文。如果在特定的网络环境中，同一 VLAN 中除了默认网关之外还有其他的网关（出口），那么路由器就会通过 Redirect 将该网关的 IP 地址告诉节点。

我们知道，当某个节点和另一个位于其他 VLAN 中的节点通信时，会将数据包临时性地发送给默认网关。默认网关的路由器收到数据包后会根据路由表选择路径，这时，如果同一 VLAN 中有另一台路由器拥有更合理的路径，就会发出一个 Redirect（类型 5）/Redirect Datagram for the Network（代码 0）的 ICMP 报文，告诉节点该网关的存在。节点收到 Redirect 报文后会在路由表中添加通往该网关的路径，下次就将数据包直接发往它的 IP 地址。

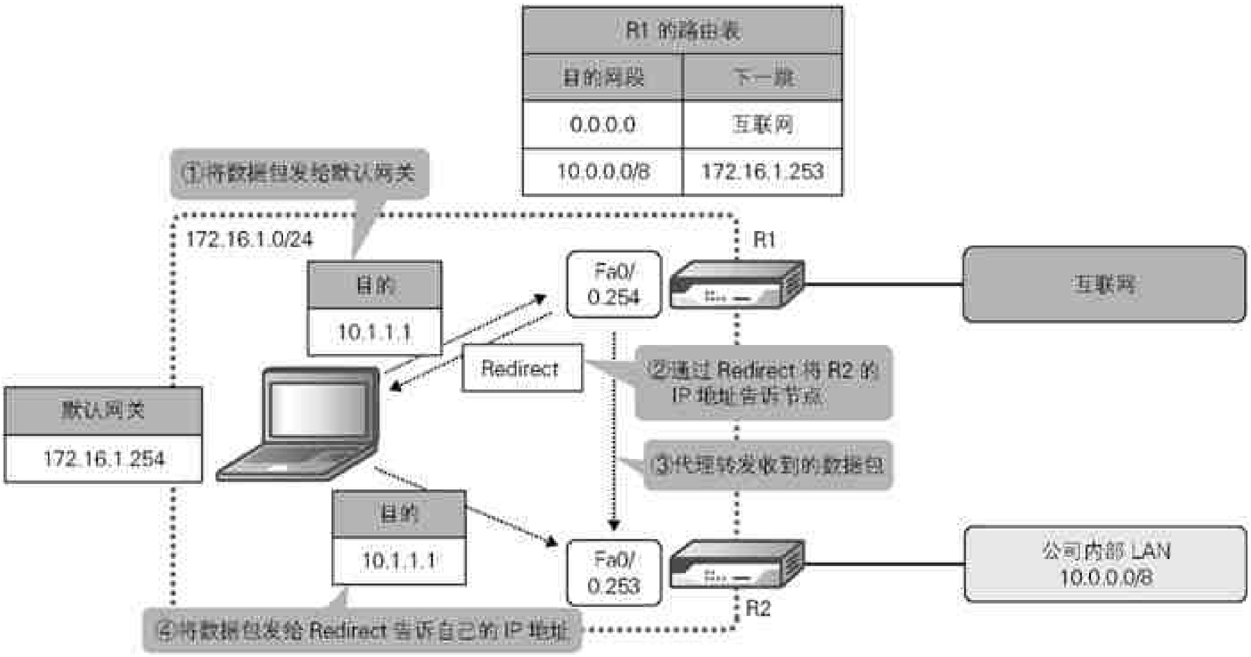


图 **2.2.59　Redirect**（类型 **5**）将最合理的网关告诉节点

通过 Redirect 可对同一 VLAN 中的路由信息进行一元化管理，非常方便。然而，并非所有设备都会返回 Redirect，有些防火墙的规格是不支持返回 Redirect 的。因此，我们在设计路由之前一定要了解相关设备的规格才行。

2.2.5.3　出现问题时先尝试用 **ping** 去排除故障

排除故障有好几种方法，需要紧急应对时，最常见的是先尝试用 ping 去排除故障。首先我们应通过 ping 确认网络层的疏通情况，确认没有问题之后再依次往上层方向走，解决传输层、会话层和应用层的问题。如果疏通情况不佳，就应该依次往下层方向走，解决网络层、数据链路层和物理层的问题。

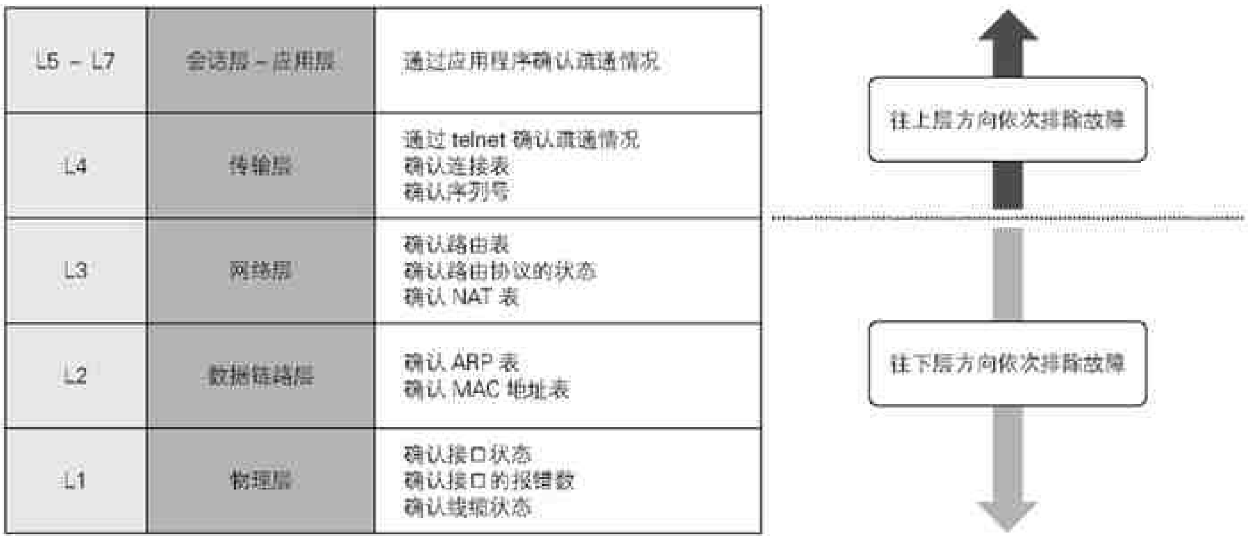


图 **2.2.60**　排除故障时人们往往会从使用 **ping** 开始

2.3 逻辑设计

前面我们学习了数据链路层和网络层的各种技术和规格（协议）。从这一节开始，我们要从实用性的角度学习如何在服务器端使用这些技术，以及在设计和架构服务器端时需要注意哪些事项。

2.3.1 整理出所需的 VLAN

首先我们来看 VLAN 设计。VLAN 设计是指对 VLAN 的逻辑结构进行设计，即将 VLAN 配置到何处，具体又如何配置。有条不紊且注重效率的 VLAN 配置对今后服务器端的管理性和可扩展性起着极大的作用，因此，我们设计的时候一定要着眼于未来。

2.3.1.1 实际所需的 VLAN 会因为诸多因素而变化

VLAN 设计的第一步是整理出所需的 VLAN。实际所需的 VLAN 取决于人们使用的设备和设备功能、数据安全、运行管理等诸多因素，我们应从不同的角度辨别具体情况，整理出真正需要的 VLAN。

首先要弄清所需的设备和功能

首先，我们应站在设备和功能的角度去考虑 VLAN。下面我们从最简单的结构出发，先来看看上层（ISP 方面）是怎样的情况（如图 2.3.1）。

- **ISP** 和 **L3** 交换机（**CE** 交换机）

  与互联网相连的 L3 交换机和路由器是根据 ISP 的指定去分配 VLAN 的。CE（Customer Edge，客户边缘）交换机因为位于 ISP 和客户的边界而得名，CE（Customer Edge）路由器的名称由来也是一样。最近好像还出现了租借形式的 ISP，如果是租借，那么此处的 VLAN 分配就都取决于 ISP，不是我们能够决定的了。VLAN 有多种分配方式，不过一般情况下人们都会在 L3 交换机外侧各安排一个，在 L3 交换机之间安排一个，在内侧安排一个（如图 2.3.1）。作为设计和架构服务器端的工程师，我们只需注意其中一个就可以了，那就是内侧的 VLAN。L3 交换机内侧的 VLAN 用于将服务器端连接到 ISP 上去 [16] 。

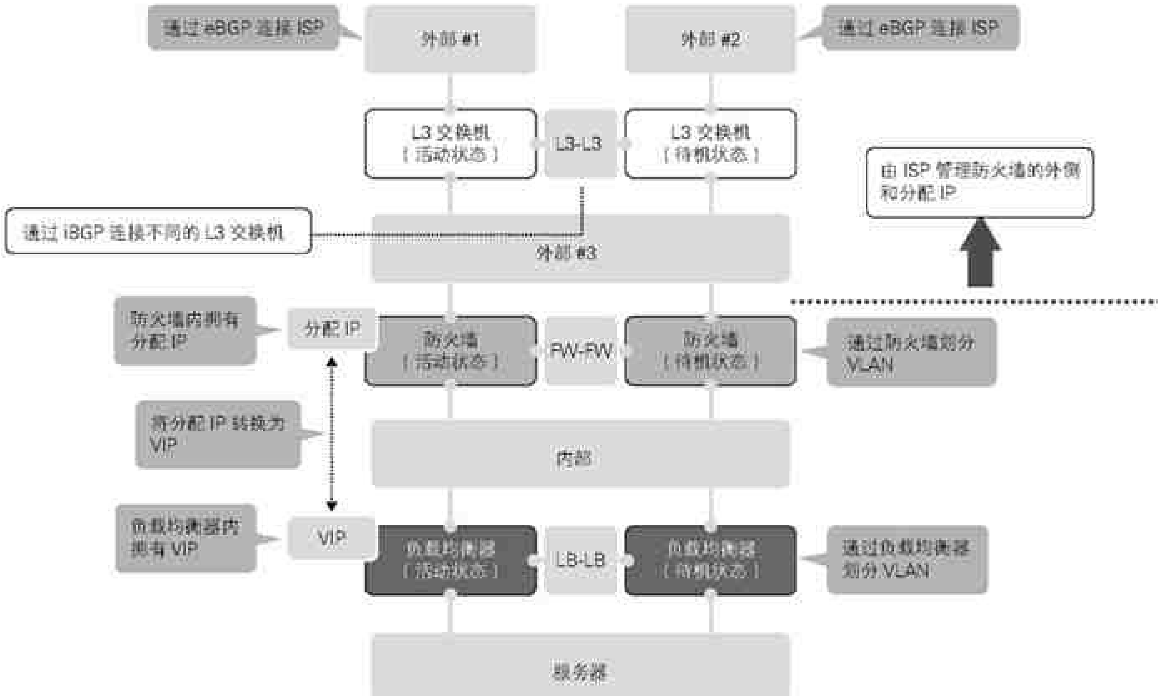


图 **2.3.1**　站在设备和功能的角度整理出所需的 **VLAN**

- **L3** 交换机（**CE** 交换机）和防火墙

防火墙外侧的 VLAN 是用来连接 L3 交换机的。一般人们会将 VLAN 划分为防火墙外侧和防火墙内侧两个部分。当然，有些设计比较特殊，同一个 VLAN（不划分 VLAN）也可以运作，笔者的确亲眼见过好几种这样的特殊结构。然而，最经典的处理办法还是将 VLAN 划分为内侧和外侧两个部分。此外，我们还需要拥有全球 IP 地址（即分配 IP 地址，由 ISP 分配）的 VLAN。分配 IP 地址的构造使其能够存在于防火墙内部，它可以转换为负载均衡器内部私网的 IP 地址。服务器端的 NAT 可能不太容易理解，因此我们还会在 2.3.4.2 节中详细解说。如果我们对防火墙作了冗余配置，那么防火墙之间可能也会需要 VLAN，通过该 VLAN 去同步机器的设置信息和状态信息。是否需要 VLAN 要看具体使用的设备，例如，思科公司的 ASA 系列是需要 VLAN 的，瞻博公司的 SSG 系列则不需要，我们在设计时务必要仔细确认好设备规格。

- 防火墙和负载均衡器

防火墙内侧的 VLAN 有多种结构，这里我们来看一下和负载均衡器连接的结构。我们可以认为负载均衡器的 VLAN 结构和防火墙

的是一样的，人们一般也是将它划分为外侧和内侧两个部分。而且，这里也需要一个能将分配 IP 地址转换为私网 IP 地址的 VLAN，该 VLAN 的形式使其能够存在于负载均衡器内部，VLAN 中会建起一个用于分散负荷的虚拟服务器。虚拟服务器的 IP 地址叫作 VIP（Virtual IP 地址，虚拟 IP 地址）。如果我们对负载均衡器作了冗余配置，那么负载均衡器之间可能也会需要 VLAN，通过该 VLAN 去同步机器的设置信息和状态信息。我们应根据实际需要适当追加。

- 负载均衡器

  最后我们来看一下负载均衡器的内侧，这是分配给服务器的 VLAN。需要划分服务器的 VLAN 时应该连接 L3 交换机，将连接负载均衡器的 VLAN 和服务器 VLAN 分隔开。而不需要划分时则应该连接 L2 交换机。

> **16** 这里举的是比较常见的 VLAN 分配例子。一般来说，ISP 除了提供服务之外还会给出如何架构和设置的实例。请遵照该实例进行操作。

从数据安全的角度整理出所需的 **VLAN**

下面我们从数据安全的角度来看 VLAN 分配。最为常见的结构是分成 Untrust（非信任区）、DMZ（DeMilitarized Zone，隔离区）和 Trust（信任区）这三个区段。区段指的是拥有同样数据安全等级的 VLAN 的群集，人们通过防火墙将不同的数据安全等级划分开来（如图 2.3.2）。

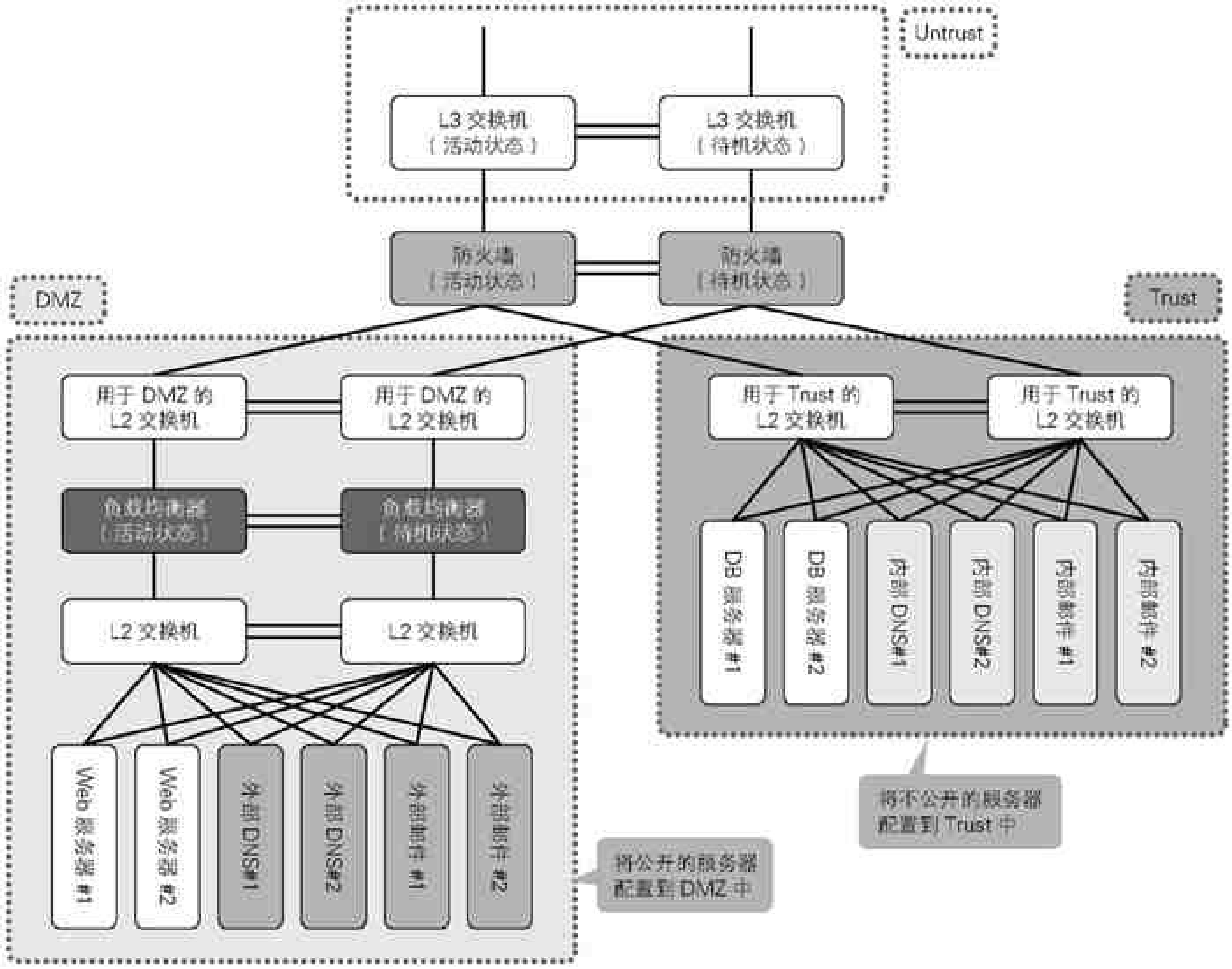


图 **2.3.2**　划分数据安全区段

DMZ 区段的安全等级比 Trust 稍低，在互联网上公开的服务器应配置到 DMZ 中，这样，当服务器遇到劫持时受害程度可降至最低。安全等级由低到高依次为 Untrust、DMZ 和 Trust。有些人有着根深蒂固的观念，认为 DMZ 只能有一个，但实际上并没有必要拘泥于一个，我们完全可以建立多个不同安全等级（微调即可）的 DMZ。整理好需要的区段之后，接下来就要为它们分配 VLAN 了。

虚拟环境中需要大量的网卡和 **VLAN**

如今，虚拟环境和系统之间早已是密不可分的关系了，而离开了网络，虚拟环境也无从谈起。由网络作后盾并且是通过网络实现的功能非常之多，其中最重要的一项就是实时迁移。实时迁移能将虚拟机的内存信息迅速地迁移到另外一台物理服务器中。用于迁移的网卡和 VLAN 和提

供服务的网卡和 VLAN 最好分开，这样能将迁移的通信流量对服务产生的影响控制在最小范围之内。

在虚拟环境中，分属于 LAN 和 DMZ 等不同 VLAN 的虚拟机共存于同一台物理服务器上。我们应在充分考虑通信流量类型和管理性的基础上，去配置虚拟交换机和虚拟机。我们也可以用一台虚拟交换机去连接所有 VLAN 的虚拟机，然而那样的话，一个 VLAN 的突发流量就会对所有 VLAN 的流量都产生影响。而且，不同虚拟交换机各自承担的职能以及它们和网卡之间的映射关系也容易变得混乱。我们不仅要将流量的影响控制在最小范围之内，还应按不同的职能将虚拟交换机分开以提高管理性。

虚拟环境让我们能够轻松地添置服务器，这样容易造成各种 VLAN 的虚拟机蜂拥而上的局面，因此考虑到系统今后的可扩展性，我们应该对网卡和 VLAN 的估算数量预留出一定的余地。

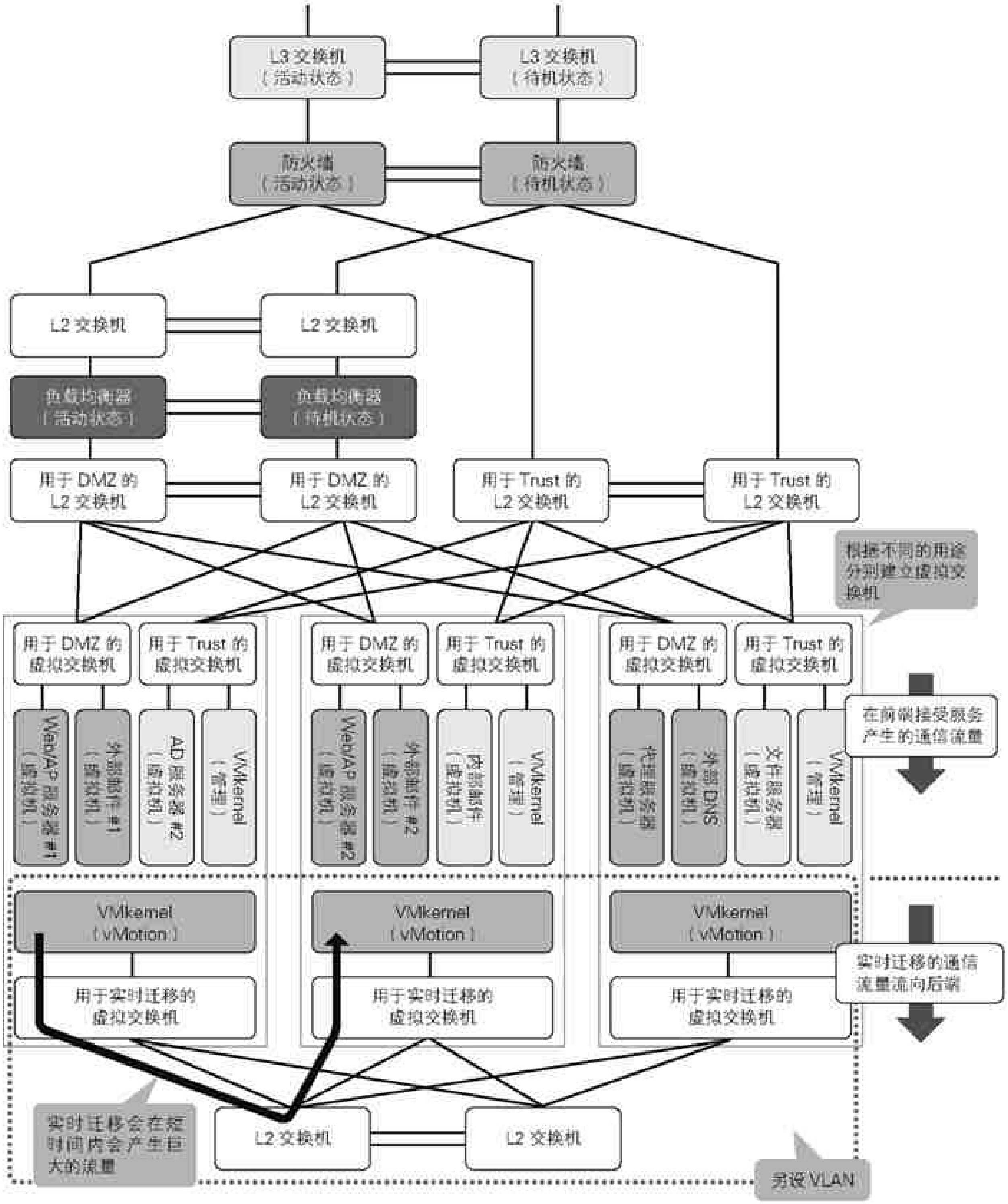


图 **2.3.3**　实时迁移的通信流量流入后端的 **VLAN**

※ 因版面关系，部分服务器在图中省略未画。

另设一个用于数据备份的 VLAN

最近，经网络转发备份数据的处理越来越多了，这种处理一般是在夜间通过批处理功能进行转发的，对正常时间段的服务没有太大影响。然而就其本质来说，它依然是在短时之内转发巨大的流量，所以比较明智的做法是另外设计一套 VLAN 和网卡来对应这种处理，将影响控制在最小范围之内。

之前讲解过的实时迁移和备份都具有相同的特点，那就是它们都会在短时之内产生巨大的通信流量并进行转发。对此我们应另设一套专门应对突发流量的 VLAN 和网卡，将突发流量对服务产生的影响控制在最小范围之内。

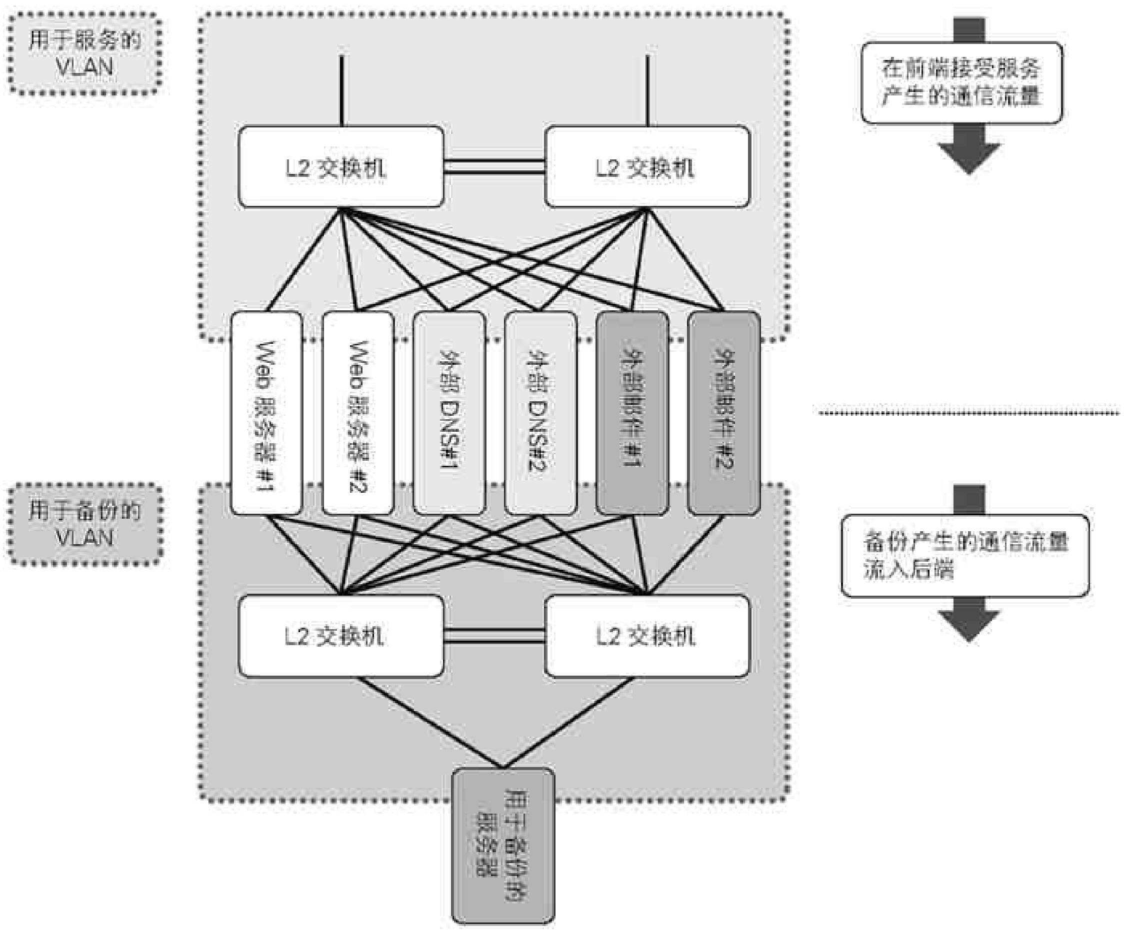


图 **2.3.4**　备份的通信流量流入后端的 **VLAN**

从运行管理的角度整理出所需的 VLAN

出于对运行管理的考虑，网络的规模越大，我们就越应该另外设计一个用于运行管理的 VLAN。通过将服务和运行管理完全分离来简化管理过程，将运行流量对服务流量产生的影响控制在最小范围之内。这时候，用于监控设备故障和工作状态的 Syslog、SNMP（Simple Network Management Protocol，简单网络管理协议）和 NTP（Network Time Protocol，网络时间协议）产生的通信流量将会经过另设的 VLAN 转发出去。Syslog、SNMP 和 NTP 将分别在 5.1.3 节、5.1.2 节和 5.1.1 节中详细说明。

最近，大多设备既配有通常使用的服务端口，也配有另外专用的管理端口，我们可将管理端口设为运行管理 VLAN。如果没有专用的管理端口，我们也可以将服务端口中的某一个端口设为运行管理 VLAN，将其作为管理端口使用。

估算 **VLAN** 的数量时应留有余地

分配给用户和服务器的 VLAN 数量是会变化的。当前需要五个 VLAN，并不意味着今后也一直只需要五个。我们应为今后的可扩展性考虑，在设计阶段留出一定的富余，将数量估算得比现状需要的多一些才行。

根据 VLAN 的位数考虑留出多少富余是个比较容易汇总的办法。假设我们当前需要五个 27 位的 VLAN，估算时可不能随随便便，拍一下脑袋就凭空决定暂时多报一个而得出共需要六个的结论。网络是逻辑的世界，决定什么事的时候一定要有充分合理的逻辑才行。回到例子，经过思考我们会发现多留三个是合理的，最后能够汇总到 24 位，让结构得以简化。

表 **2.3.1**　估算 **VLAN** 的数量时应留有余地

十进制写法	255.255.255.0	255.255.255.128	255.255.255.192	255.255.255.224	用途
斜线写法	/24	/25	/26	/27	
最大 IP 数量	254（=256-2）	126（=128-2）	62（=64-2）	30（=32-2）	
分配网段	192.168.1.0	192.168.1.0	192.168.1.0	192.168.1.0	用户 VLAN①
				192.168.1.32	用户 VLAN②
			192.168.1.64	192.168.1.64	用户 VLAN③
				192.168.1.96	用户 VLAN④
		192.168.1.128	192.168.1.128	192.168.1.128	用户 VLAN⑤
				192.168.1.160	留待日后扩展
			192.168.1.192	192.168.1.192	留待日后扩展
				192.168.1.224	留待日后扩展

2.3.1.2 规定 VLAN 的 ID

人们是通过 VLAN ID（数字）来识别 VLAN 的。VLAN ID 在交换机内部设置，负责将不同的 VLAN 区别开。我们知道，只要不用打标 VLAN，VLAN 就仅在一台交换机内有效，因此，即使是同一 VLAN 我们也可以基于交换机为它们设置不同的 VLAN ID，并且能够保证通信正常。如下图所示，我们看到右边的交换机已经将 VLAN 设成了 VLAN2，于是决定在左边的交换机中将 VLAN 设成 VLAN1，这样做也不是不行的。然而，站在运行管理的角度去看，就会发现这种设置会带来极大的麻烦，那就是我们还得花点时间在脑子里给它们一一对上号才行，既费力又耗时。因此，对于同一 VLAN，我们应该统一设置相同的 ID，在设计阶段就将 VLAN ID 的基本规则明确下来。

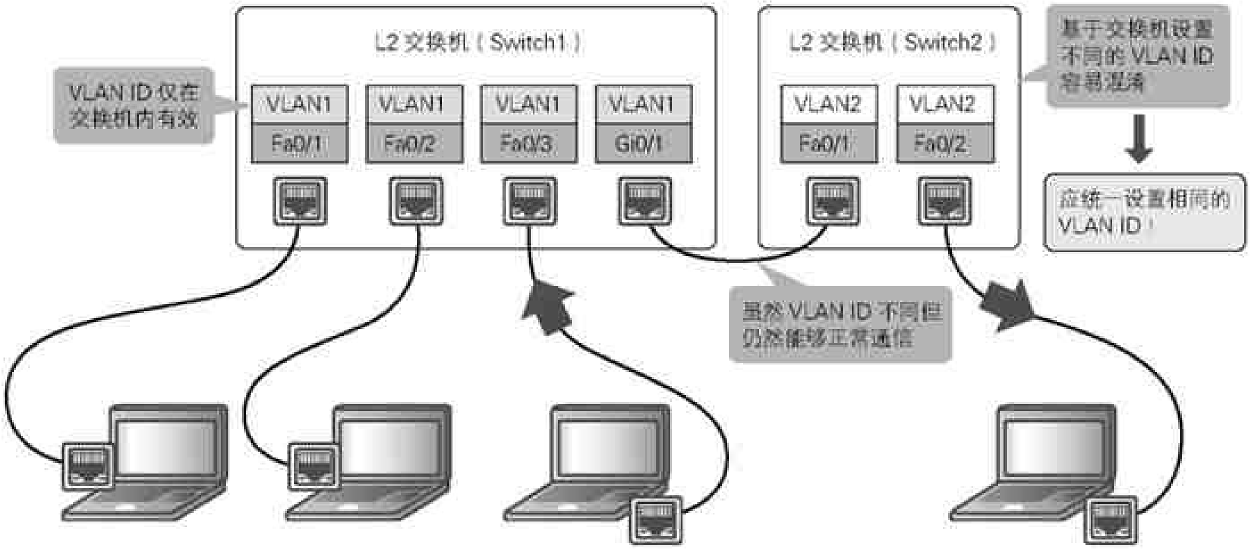


图 **2.3.5　VLAN ID** 不统一的话很容易混淆

使用打标 VLAN 时，则务必要保持 VLAN ID 一致，本征 VLAN 也是如此。默认的本征 VLAN 是 VLAN1，所以我们也可以直接使用它。不过，使用默认值时存在数据安全风险。有一种网络攻击叫作“VLAN 跳跃”[17]，是利用 VLAN 的标签规格进行恶意攻击的，如果使用默认值就会成为被攻击的对象。所以我们最好不要使用默认的 VLAN1，而是对其稍作修改，这样才更加安全。

[17] 此种攻击会发送一个打上了 VLAN 标签的报文，跨越访问控制进行 DoS 攻击。

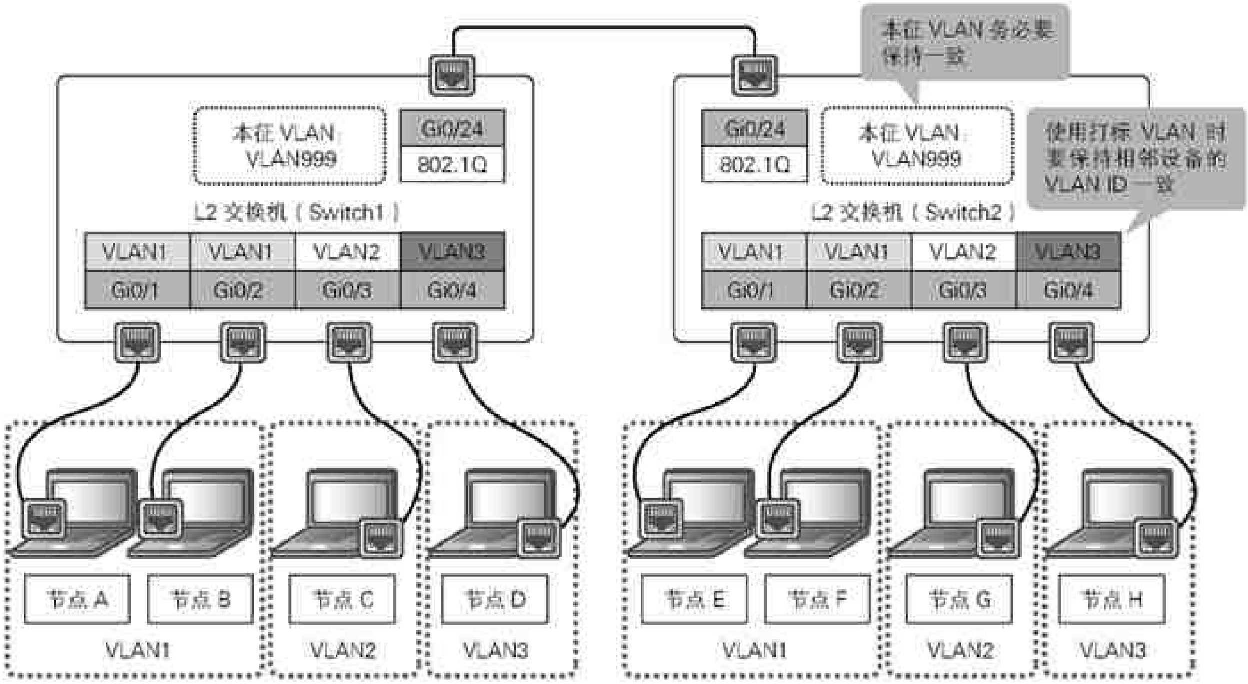


图 **2.3.6**　使用打标 **VLAN** 时要保持本征 **VLAN** 一致

我们也可以为 VLAN 命名。起一个无论是谁都能看明白的名字对今后的运行管理有着重大的意义。管理人员不可能一直固守同一岗位，因此我们应制定一个简单易行的命名基本规则，确保其他人员接手工作时能够很快理解名字的含义。

2.3.2　在考虑数量增减的基础上分配 IP 地址

下面我们来看 IP 地址设计。IP 地址设计指的是给配置好的 VLAN 分配怎样的网络地址。有条不紊且注重效率的 IP 地址分配对今后的管理性和可扩展性起着极大的作用，因此，我们设计的时候一定要着眼于未来。

2.3.2.1　IP 地址的估算数量应高于当前所需数量

IP 地址设计的第一步是弄清所需 IP 地址的数量。实际所需的 IP 地址取决于人们使用的设备和设备功能、服务器数量、客户端数量等诸多因素，我们应从不同的角度辨别具体情况，整理出架构环境时真正需要的 IP 地址。

估算时应留有余地

如果是在 IP 地址绝对不会增加或减少的 VLAN，例如防火墙之间或负载均衡器之间的 VLAN 等，我们只要数清楚 IP 地址的数量就好了。

然而事实并没那么简单，对分配给用户和服务器的 IP 地址我们必须多估算一些才行。如果考虑得过于简单，比如单纯地认为“需要 10 个 IP 地址嘛，那我用 28 位去分配就好了”，那么今后需要更多 IP 地址的时候，我们就不得不重新分配 VLAN，非常麻烦。新增 VLAN 相当于新设路由器，是一件十分耗时费力的事情，所以我们应合理预估今后 IP 地址的增加情况，尽可能多估算一些数量。

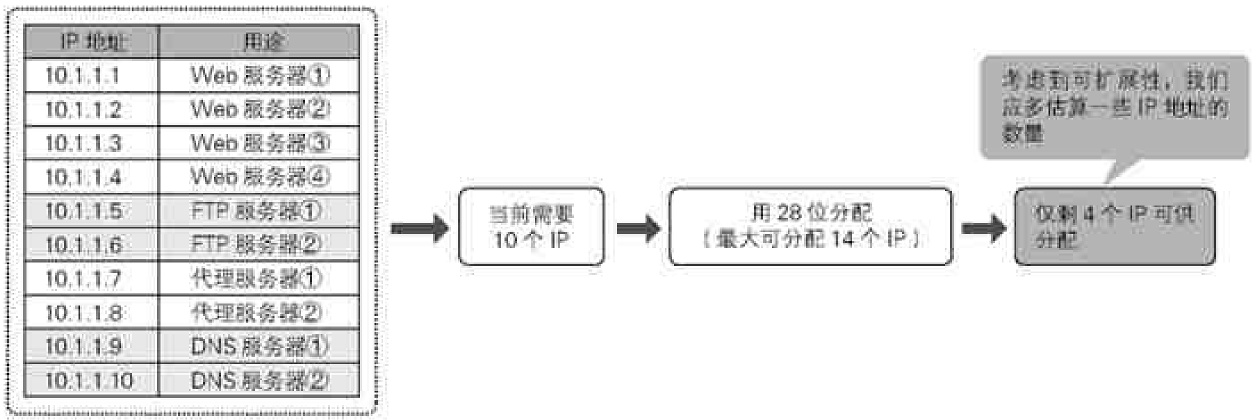


图 **2.3.7**　估算数量应高于当前所需数量

估算时不要忘了用于特殊用途的 **IP** 地址

如果我们给服务器和网络设备作了冗余配置，那么除了物理 IP 地址之外它们一般还需要一个共享 IP 地址。总得来说就是，在考虑 IP 地址的数量时，不要忘记我们需要的有活动机的 IP 地址、备用机的 IP 地址和二者共享的 IP 地址 [18] 。共享

[18] 有些设备没有共享 IP 地址，而是直接使用活动机的物理 IP 地址，这取决于我们选用的机器规格。因此在设计时请一定要好好确认这点。

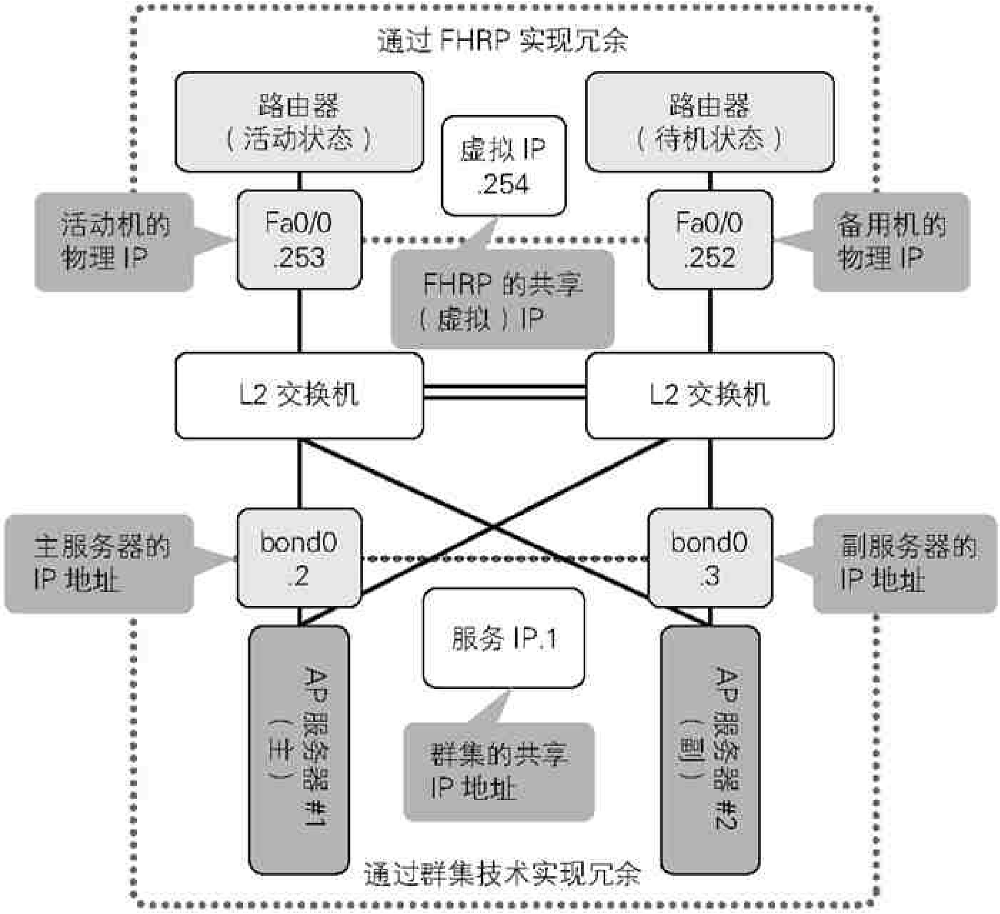


图 **2.3.8**　作冗余配置时所需的 **IP** 地址会增加

VLAN 是要做大还是做小

网络是要做大还是做小，这对运行管理有着重大的影响。

以往人们一直推崇按 IP 地址所需个数将网络尽量做小的思路，然而，现在这样做已经未必是最好的选择了。实际上，将配置在 LAN 内的用户 VLAN 用 21 位分隔开，在 1 个 VLAN 中囊括数千台终端的事例也不是没有。当然，那样的话同一 VLAN 中的终端都会收到本身并不需要的广播，不过最近的 OS 已经不像过去那样频繁地使用广播了，接口的速度也比以前快了很多，所以网络即使做得很大也完全可以顺畅地运行。网络越大 VLAN 就越少，设置和运行管理也就越轻松。

将网络做大时有一点必须注意，那就是桥接环路。网络越大，死循环的影响范围就越大，一旦发生桥接环路，那么所有隶属于该 VLAN 的终端就会无法通信。不仅如此，有些路由点上的所有终端也可能会无法通

信。因此，我们在设计时，务必要用 STP（Spanning Tree Protocol，生成树协议）防止死循环的发生。桥接环路和 STP 将在 4.1.2.1 节中详细说明。

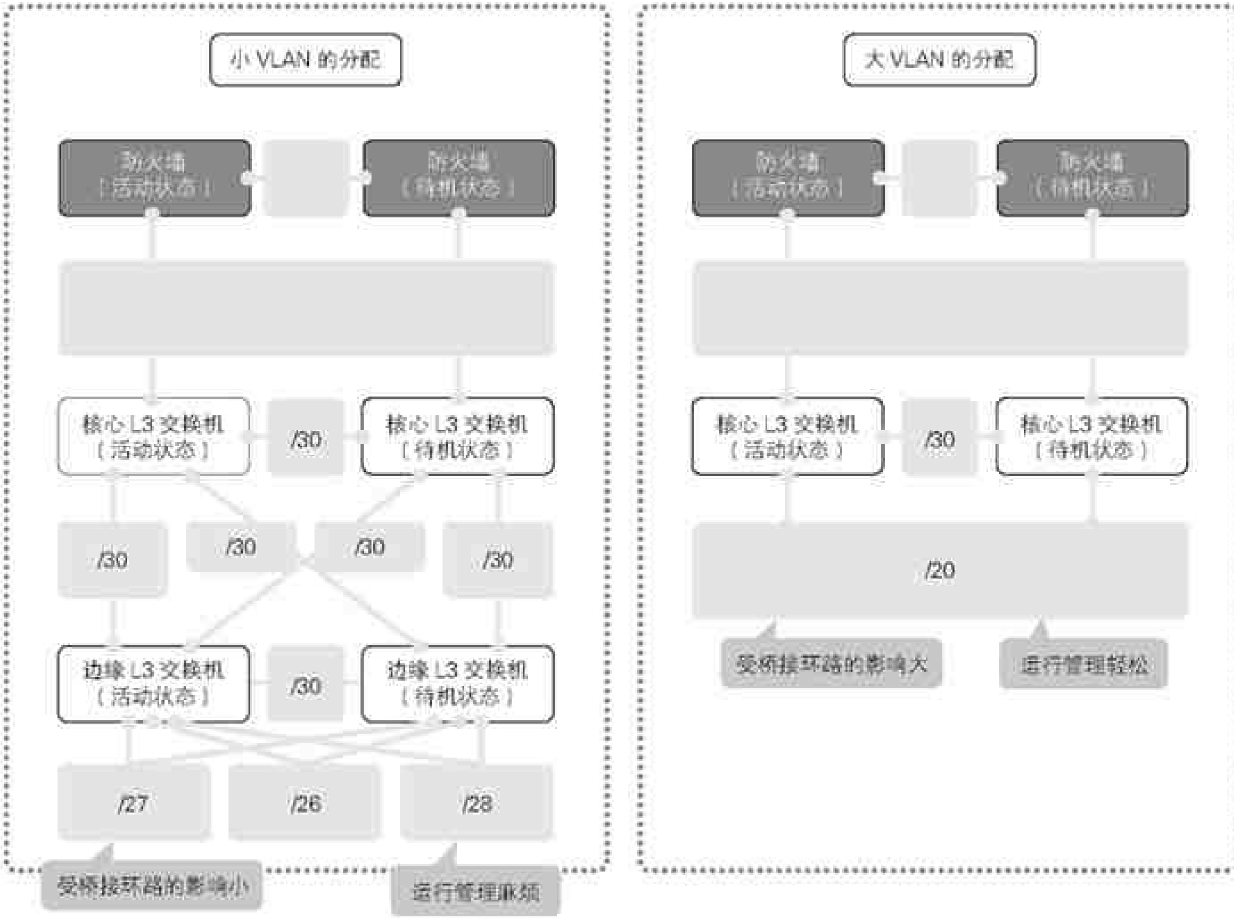


图 **2.3.9　VLAN** 是要做大还是做小

关于网络的设计笔者还要提一点，那就是无论需要多少个 IP 地址都用“/24”去统一分隔的手法。用 24 位去划分比较容易理解，也便于管理。网络要做到多大取决于客户的需求，我们应看准和认清需求之后再去考虑网络的具体设计。

2.3.2.2　按顺序排列网段，使之更容易汇总

注重效率的子网划分对路由汇总起着良好的作用，有利于简化路径。我们在设计的时候应按一定的顺序排列网段，使它们更加容易汇总、更加简约明朗。预先定义好在哪里汇总、用多少位去汇总，也有助于提高系统今后的可扩展性。

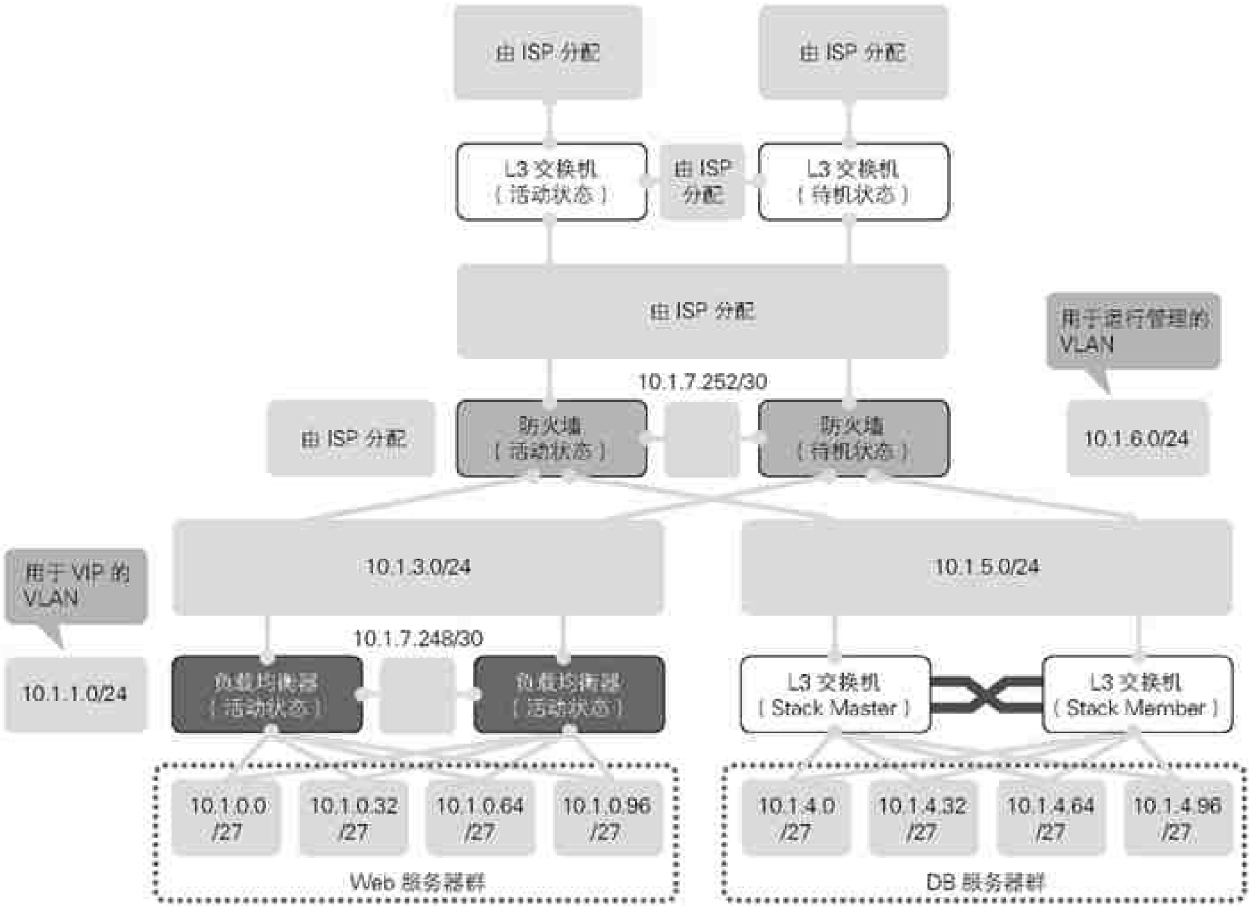


图 **2.3.10**　将网段按顺序排列

笔者一般会将网段整理成一张表格（如表 2.3.2）以便随时查询。整理出一张这样的表格后，网段的汇总和管理都会更加轻松。下面这张表是对网段分配的整理结果，在将网段分配给服务器的过程中，最终要达到能够用 24 位对网段进行划分和汇总的结果。做这种 IP 地址设计就像玩俄罗斯方块一样，充满了乐趣。

表 **2.3.2**　将网段整理成表

十进制写法	255.255.255.0	255.255.255.128	255.255.255.192	255.255.255.224	255.255.255.240	255.255.255.248	255.255.255.250	用途
斜线写法	/24	/25	/26	/27	/28	/29	/30	
最大 IP 数量	254（=256-2）	126（=128-2）	62（=64-2）	30（=32-2）	14（=16-2）	6（=8-2）	2（=4-2）	
分配网段	10.1.0.0	10.1.0.0	10.1.0.0	10.1.0.0				Web①
				10.1.0.32				Web②
			10.1.0.64	10.1.0.64				Web③
				10.1.0.96				Web④
		10.1.0.128	10.1.0.128	10.1.0.128				留待日后扩展
				10.1.0.160				留待日后扩展
			10.1.0.192	10.1.0.192				留待日后扩展
				10.1.0.224				留待日后扩展
	10.1.1.0							VIP
	10.1.2.0							留待 VIP 日后扩展
	10.1.3.0							FW-LB 之间
	10.1.4.0	10.1.4.0	10.1.4.0	10.1.4.0				DB①
				10.1.4.32				DB②
			10.1.4.64	10.1.4.64				DB③
				10.1.4.96				DB④
		10.1.4.128	10.1.4.128	10.1.4.128				留待日后扩展
				10.1.4.160				留待日后扩展
			10.1.4.192	10.1.4.192				留待日后扩展
				10.1.4.224				留待日后扩展
	10.1.5.0							FW-L3 之间
	10.1.6.0							运行管理
	10.1.7.0	10.1.7.0						空置
		10.1.7.128	10.1.7.128					空置
			10.1.7.192	10.1.7.192				空置
				10.1.7.224	10.1.7.224			空置
					10.1.7.240	10.1.7.240		空置
						10.1.7.248	10.1.7.248	LB-LB 之间
							10.1.7.252	FW-FW 之间

2.3.2.3 必须统一规定从何处开始分配 IP 地址

网段分好之后，接下来我们应该在其中的何处开始分配 IP 地址，具体又该如何分配呢？对于这些问题也都做出统一规定的话，理解起来就会

易如反掌。例如，我们可以做出这样的分配规定：服务器和用户终端的IP 地址从数字最小的编号用起，网络设备的 IP 地址则从数字最大的编号用起。这样，IP 地址的分配和使用就有了统一性，后面的管理也就方便多了。此外，我们在设计类似的结构时也可以照葫芦画瓢地采用同样的基本规则，那样也会轻松很多。只要没有现成的基本规则，对于类似的结构，笔者总是会制定一模一样的基本规则以提高作业效率、减少作业失误。

IP 地址	用途
10.1.1.0/24	网络地址
10.1.1.1	Web 服务器①
10.1.1.2	Web 服务器②
10.1.1.3	Web 服务器③
10.1.1.4	Web 服务器④
10.1.1.5	留待 Web 服务器日后扩展
10.1.1.6	留待 Web 服务器日后扩展
10.1.1.7	留待 Web 服务器日后扩展
10.1.1.8	文件服务器①
10.1.1.9	文件服务器②
10.1.1.10	文件服务器③
10.1.1.11	留待文件服务器日后扩展
~	
10.1.1.252	L3 交换机物理 IP（备用机）
10.1.1.253	L3 交换机物理 IP（活动机）
10.1.1.254	L3 交换机共享 IP
10.1.1.255	广播地址

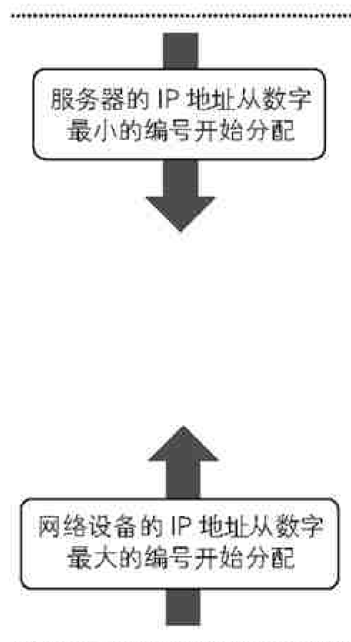


图 **2.3.11**　统一分配 **IP** 地址的基本规则

2.3.3　路由选择以简为上

下面，我们来看路由设计。路由设计是指设计在什么地方以及如何对分配好的网络地址进行路由选择。在哪里、如何进行路由选择对今后的管理性和可扩展性起着极大的作用，因此我们设计的时候一定要着眼于未来。

2.3.3.1　考虑在路由选择中使用哪些协议

首先，我们要确定在路由选择中使用哪些协议。以往，人们将 Apple Talk、SNA（Systems Network Architecture，系统网络架构）、FNA（Fujitsu Network Architecture，富士通网络架构）和 IPX 等多种协议混合在一起使用，每一个协议都得进行路由选择。不过，现在大多数环境都只有 IP，和以往相比简单了不少。大多数情况下，人们仅将 IP 数据包视为路由选择的对象。

2.3.3.2 考虑采用哪种路由选择方法

然后，我们应该确定采用哪种路由选择方法。路由选择有两种方法，分别是静态路由选择和动态路由选择。

用于互联网的通信采用默认路由

当我们的通信对象在互联网上时，其 IP 地址是不特定的。因此，对于在网络中扮演着通往互联网的出口这一角色的设备，比如防火墙和负载均衡器等，我们应将它们的 IP 地址设为默认网关。如果我们对这些设备作了冗余配置，那么其共享 IP 地址就是默认网关。

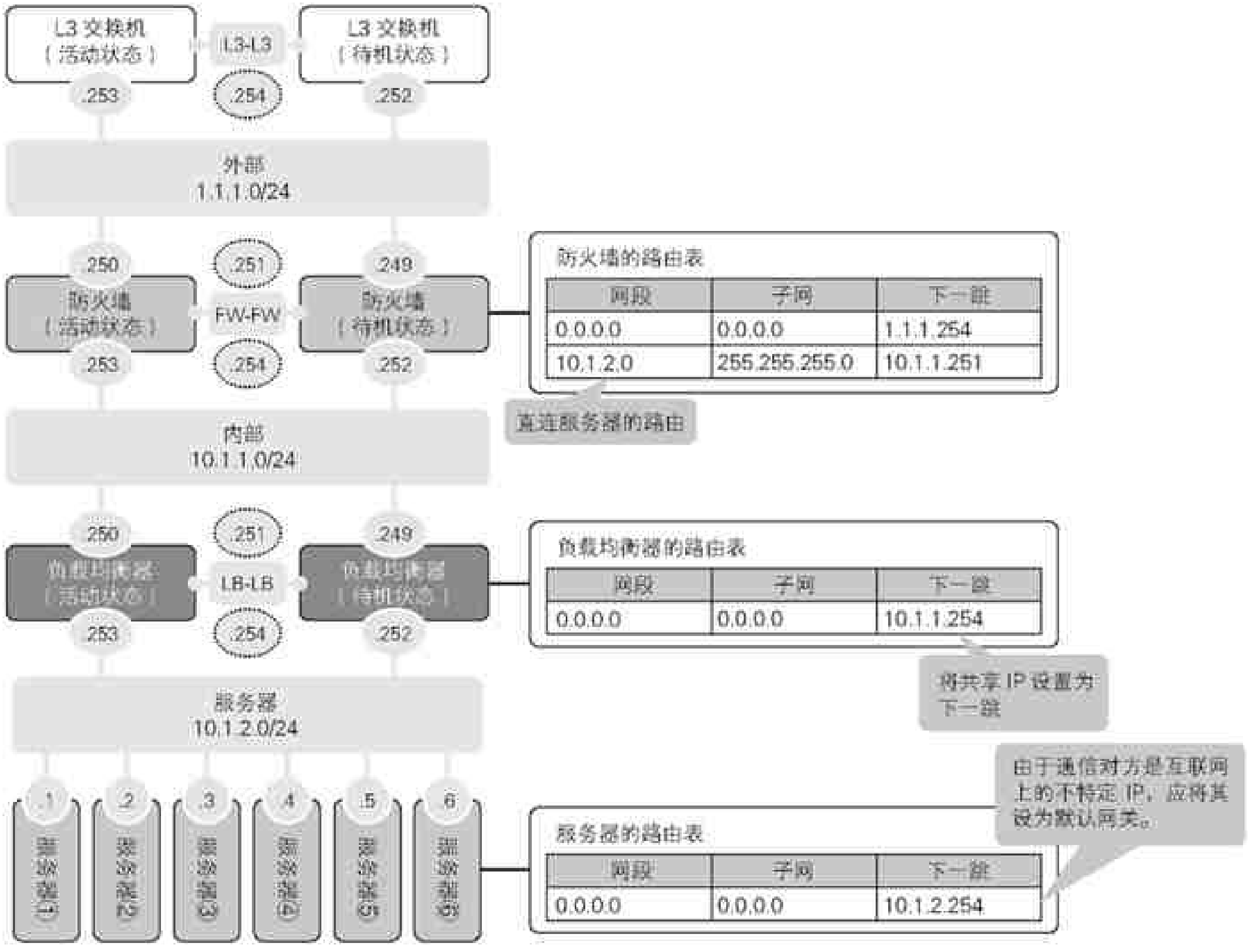


图 **2.3.12**　默认网关为共享 **IP** 地址

※ 实际上 Connected（直连的）网段也会写入路由表。这里为了方便大家理解，仅画出了设置的路径信息。

如果公开服务器的后端有配置了 DB 服务器和 AP 服务器的 VLAN，我们就要对公开服务器设置能够抵达该 WLAN 的静态路由。偶尔可能会出现后端也设置了默认网关，结果导致一个服务器拥有好几个默认网关的情况，但这是错误的做法，服务器的默认网关只能有一个。如果通往特定 VLAN 的路径有别的下一跳，我们也应设置能够抵达该 VLAN 的静态路由。

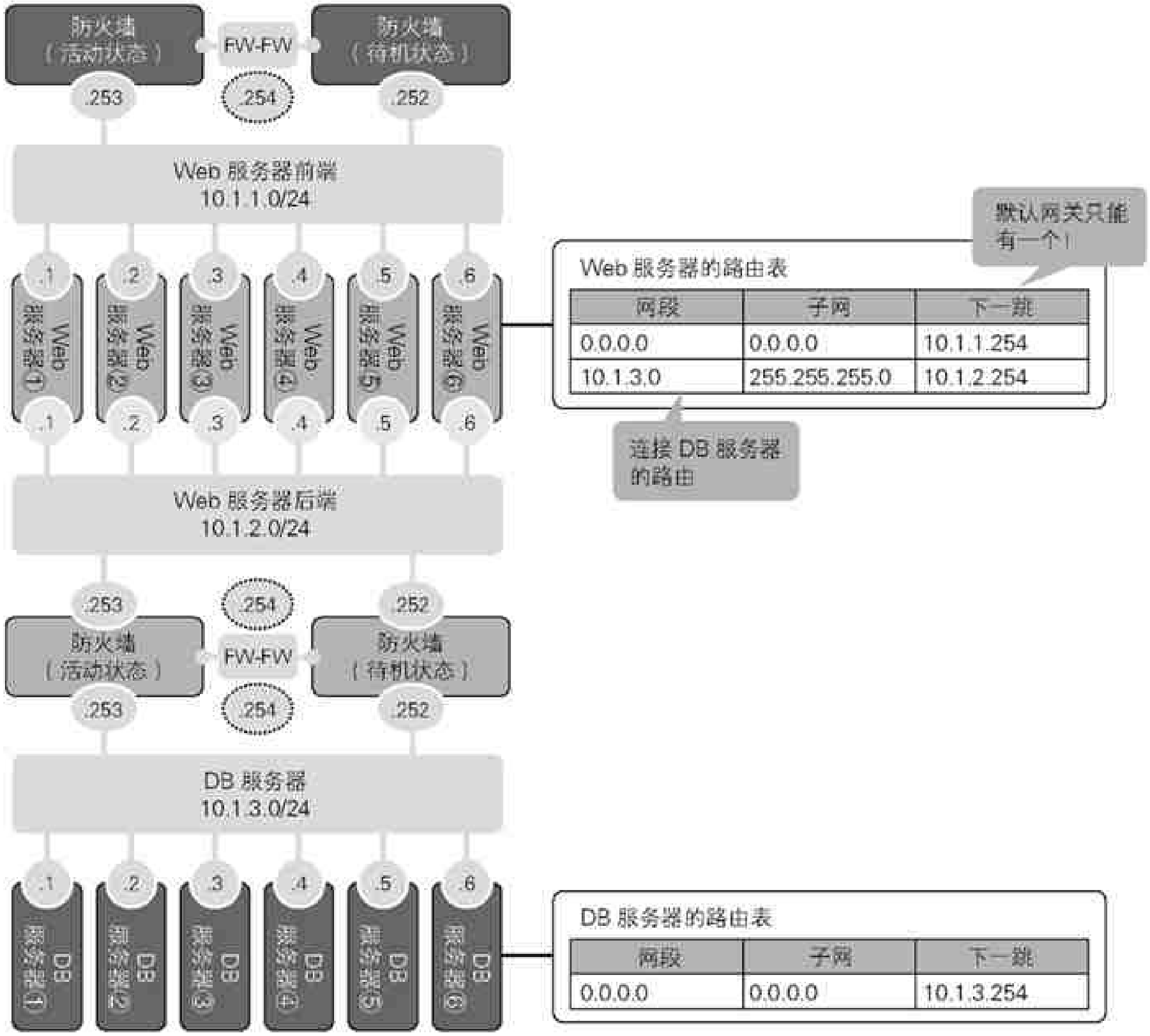


图 **2.3.13**　默认网关只能有一个

和 **ISP** 的连接点上运行着 **ISP**

ISP 上运行着 BGP。位于系统最上层的 L3 交换机在 ISP 和 eBGP 对等体之间以及 L3 交换 机之间安排 iBGP 对等体，以此来确保从互联网通向服务器端的路径 [19] 。而且，这个时候会通过 BGP 属性控制使用的线路。通向互联网的路径是将 ISP 的 CE（Customer Edge，客户边缘设备） 交换机作为下一跳的默认路由 [20] 。

> [19] 实际上是通过将静态路由和 IGP 再发布给 BGP，或是仅将特定的路径写入 BGP 表来确保路径。

[20] 不同的 ISP 和服务会有不同的上层 BGP 结构。本书中讲解的 BGP 结构均以单一 ISP 双线接入（冗余配置）为前提。

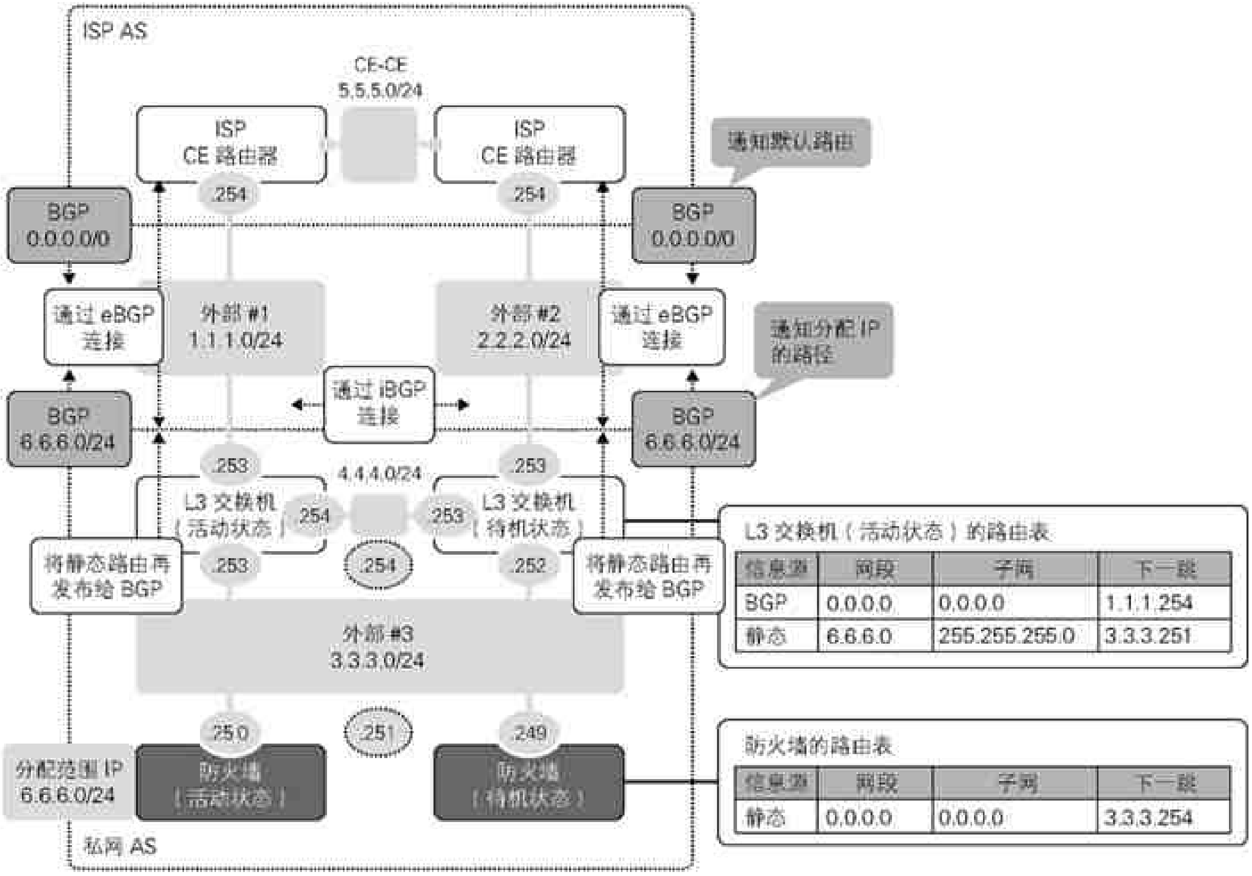


图 **2.3.14**　**ISP** 是通过 **BGP** 接入的

※ FW-FW 之间的 VLAN 因和 ISP 接入无关，所以图中省略未画。

统一路由协议

在 LAN 内运行路由协议时，将各种路由协议统一成一个比较合理。当然，我们也可以通过再发布让几个不同的路由协议同时运行，事实上也的确存在非这样处理不可的情况。但是，考虑到今后的运行管理，还是将它们迁移到一个路由协议上的做法更加合适。

路由选择方式分核心路由选择和边缘路由选择两种

路由选择方式也是设计 LAN 内路由选择的一个关键。将路由选择点（让数据包选择路径的点）落实到网络结构的哪个阶层，决定了路由选

择方式是核心路由选择还是边缘路由选择。

核心路由选择是由位于网络中心的核心交换机为所有路由数据包选择路径的方式，这种方式下要给核心交换机配置 L3 交换机、给边缘交换机配置 L2 交换机。路由选择点因为只有核心交换机这一处，所以该方式操作方便，故障也容易排除。不过，如果一个 VLAN 内发生了桥接环路，影响就会波及整个 VLAN，所以我们务必认真对待 STP 设计。此外，由于所有的路由数据包都会经过核心交换机，我们还要注意核心交换机的处理负荷问题。

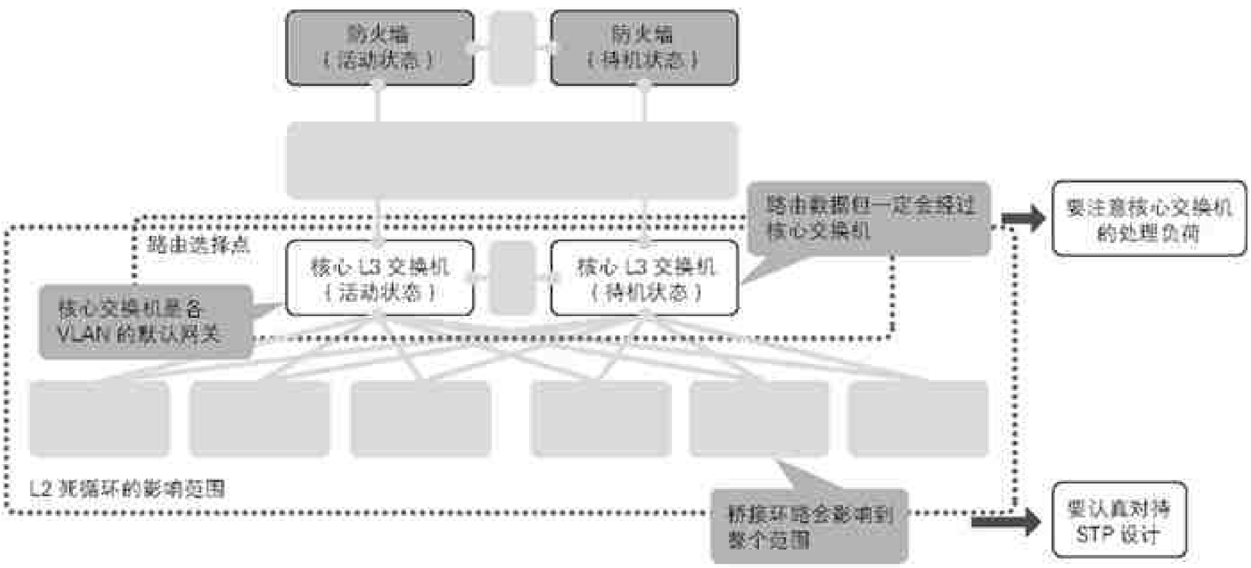


图 **2.3.15**　核心路由选择方式仅靠核心交换机来选择路径

边缘路由选择是由位于网络边缘的边缘交换机选择路径的方式，这种方式下要给核心交换机和边缘交换机都配置 L3 交换机。核心交换机和边缘交换机之间通过路由协议同步 LAN 内路径，因此边缘交换机也能够进行路由选择。同属一台边缘交换机的 VLAN 之间的路由数据包只需经该边缘交换机处理即可，因此负荷得以分散，而且即使发生桥接环路，其影响也不至于扩散到整个范围。不过这种方式也有缺点，比如路由选择点较多、构造稍嫌复杂、运行管理也会更难一些。

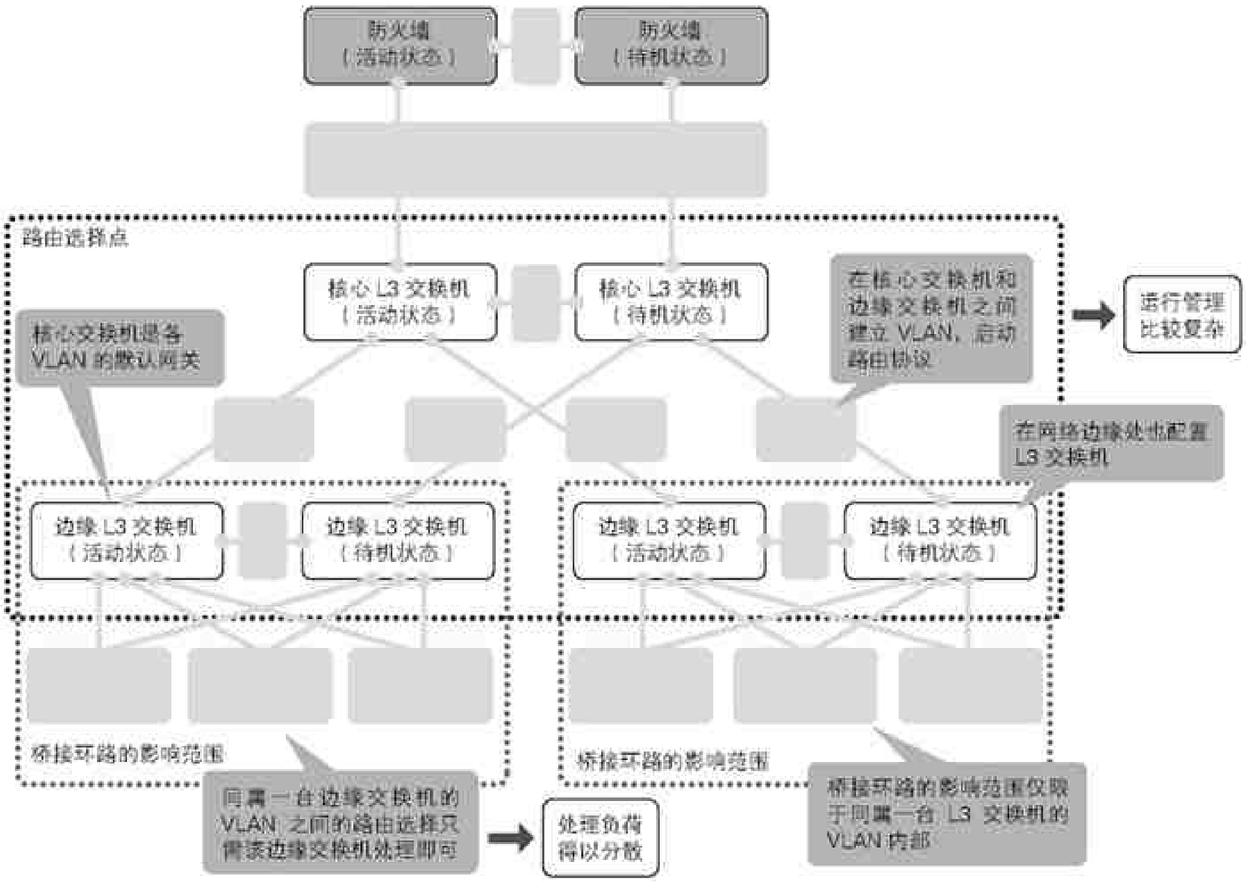


图 **2.3.16**　在边缘路由选择方式中，边缘交换机也能够选择路径

过去人们比较倾向于将 VLAN 尽量做小且采用边缘路由选择的方式，然而现在主流已经变成了将 VLAN 尽量做大且采用核心路由选择的方式，这是因为后者能将运行管理方面的优势最大化。两种路由选择方式都有各自的优点和缺点，选择哪一种取决于客户的需求。我们应认清需求之后再开始设计。

表 **2.3.3**　核心路由选择和边缘路由选择都有各自的优点和缺点

路由选择方式	核心路由选择	边缘路由选择
路由选择点	核心交换机	核心交换机 ～ 边缘交换机
各 VLAN 的默认网关	核心交换机	边缘交换机

逻辑结构	简单	稍嫌复杂
运行管理	简单	稍嫌复杂
路由数据包的处理负荷	全部集中在核心交换机上	也会分散到边缘交换机上
L2 死循环的影响范围	大	小

让路由器和 **L3** 交换机处理路由协议

防火墙、负载均衡器和服务器也是可以启动路由协议的，不过笔者并不推荐这样做。我们偶尔会遇到态度强硬的管理人员翻出不知是从哪里找来的使用手册，说既然有这项功能那就要用用看，这种想法是不对的。“有这项功能”和“用这项功能”是完全不同的两码事，尤其在路由协议方面，我们还是交给专用设备去处理比较合适。因为即使通过防火墙、负载均衡器或服务器启动了路由协议，我们并不能保证路由协议今后也能够长期稳定地运作。所以，路由协议还是交给 L3 交换机和路由器去处理吧。

2.3.3.3　将路径汇总以减少路径数量

我们应该确定在什么地方、将多少条路径汇总起来。任凭路径发展的话，其数量只会有增无减，也会越来越乱，还有可能引发设置失误。所以我们要按顺序分配网络地址，合理高效地汇总路径。

图 2.3.18 所示的是一个利用防火墙将 /27 汇总到 /24 中，减少路径数量的示例。

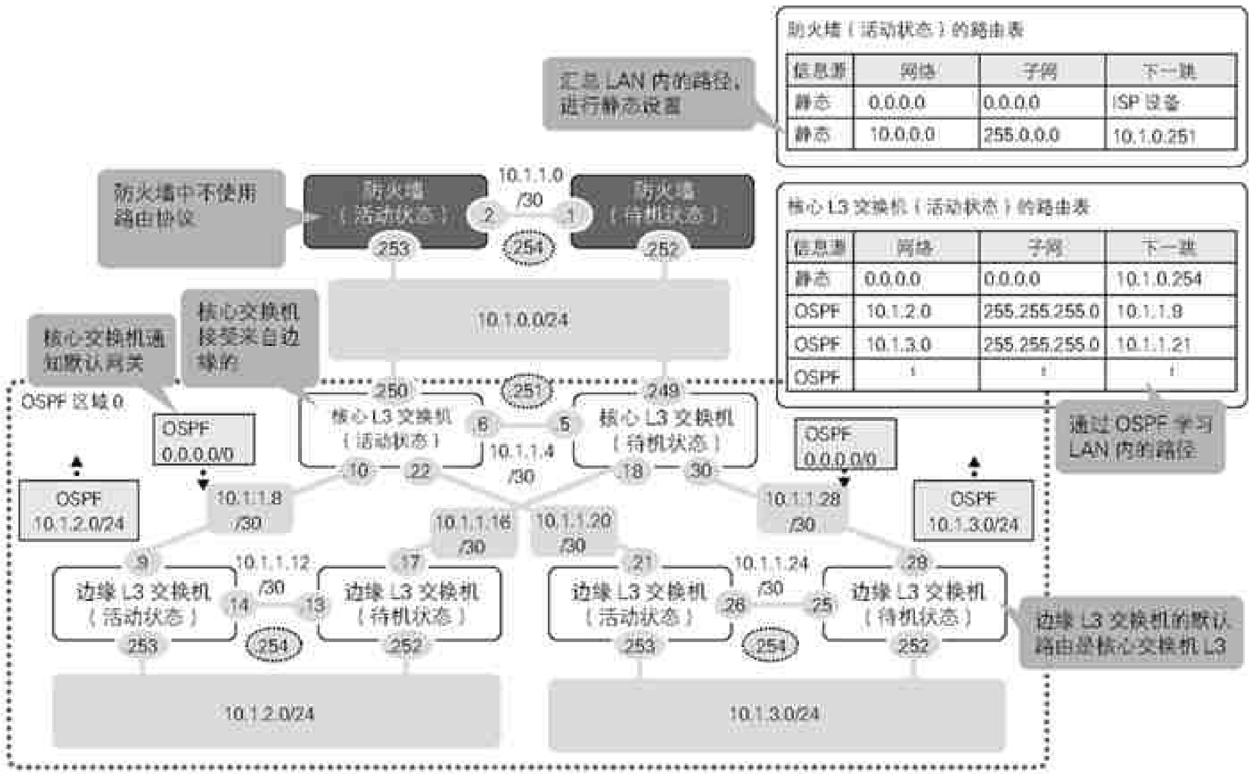


图 **2.3.17**　要使用路由协议就应该准备 **L3** 交换机和路由器

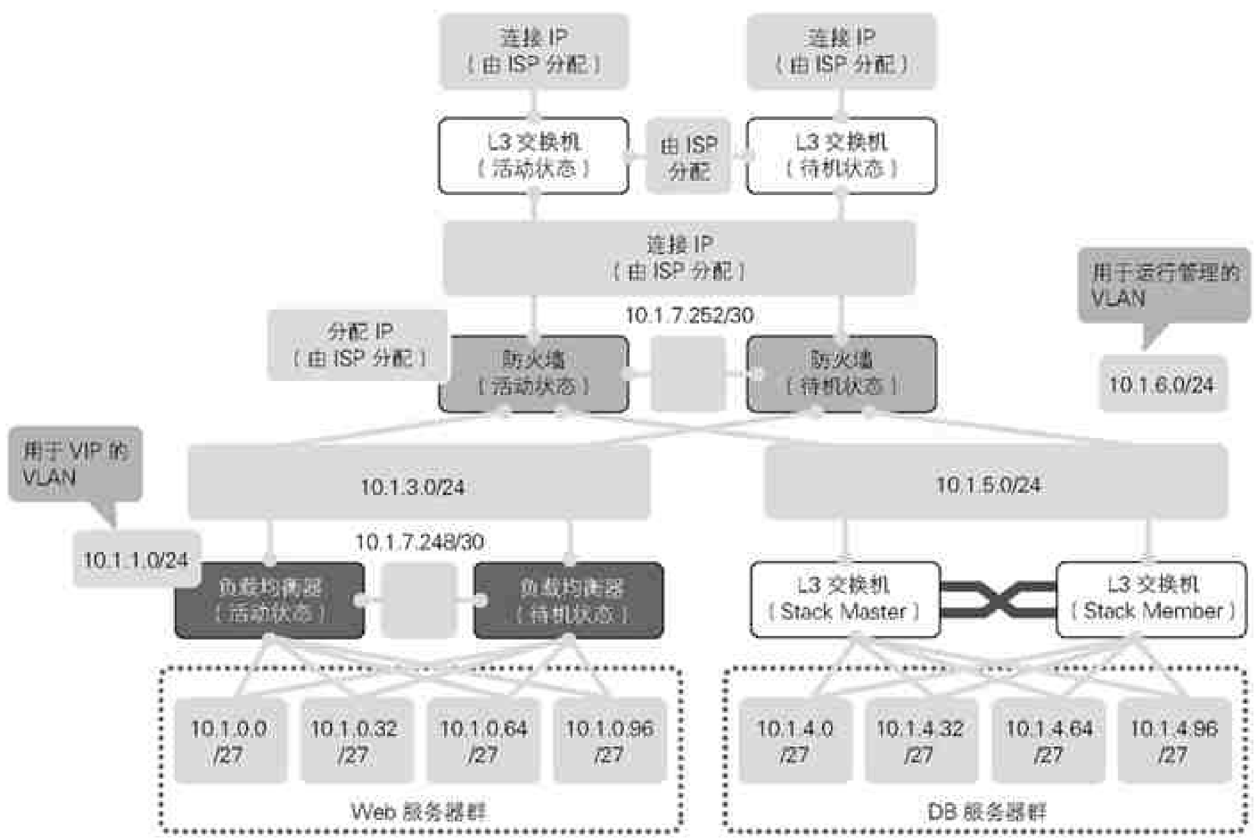


图 **2.3.18**　通过汇总来简化路径

2.3.4　NAT 要按入站和出站分别考虑

不了解 NAT 的人会觉得它很复杂，不知不觉就对它敬而远之。然而，当我们规定好在哪里转换地址之后，再去考虑起点和终点应该分别安排在何处，又应该如何进行地址转换的话，事情就没那么复杂了。本书将按入站（由互联网通向服务器端的通信）和出站（由服务器端通向互联网的通信）来讲解 NAT[21] 。

> [21] 其实负载均衡也是一种利用了地址转换的技术，只是放在这里一起介绍容易混淆，所以将其放在第 3 章中详细说明。

2.3.4.1　NAT 是在系统边界进行的

和路由选择一样，NAT 的设计也是以简为上的。如果在一个系统的很多地方都进行了地址转换，我们就会渐渐地理不出头绪来，所以最好还是尽量避免重复的地址转换，采用单纯明朗的设计。

NAT 是在系统的边界进行的。对服务器端来说，它和互联网的边界是防火墙，因此 NAT 多在防火墙中进行，即在防火墙中将全球 IP 地址转换为私网 IP 地址。

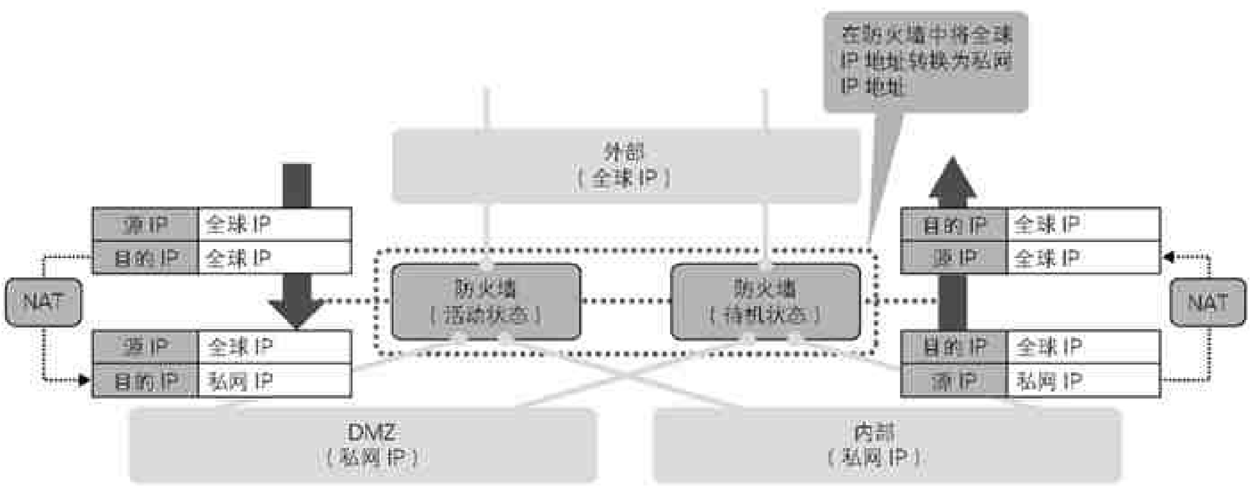


图 **2.3.19**　**NAT** 多在防火墙中进行

2.3.4.2　通过入站通信转换地址

入站（由外部进入内部的）通信是针对公开服务器的通信，需要转换目的 IP 地址。其原理是，先找出需要在互联网上公开的服务器，然后对该服务器的 IP 地址进行转换。

服务器端的 NAT 比较特殊，不太容易弄懂，下面我们来整理一下。

ISP 和全球 IP 地址签约之后会提供两个全球 IP 地址范围，一个用于连接，另一个则用于分配 [22] 。用于连接的 IP 地址范围是指用于连接 ISP 的多个 VLAN，在防火墙的 Untrust 区段中设置；用于分配的 IP 地址空间则是指用于公开服务器和出站 NAPT 的 IP 地址的 VLAN，仅存在于防火墙内部。ISP 将分配 IP 地址范围的下一跳设为防火墙的连接 IP 地址 [23] ，这样，发送给分配 IP 地址范围的数据包就会进入防火墙而不是代理 ARP，防火墙在 NAT 表中查询到该数据包的目的 IP 地址后将其转换为私网 IP 地址，然后将数据包发送给公开服务器。

> [22] 有些 ISP 将用于连接的 IP 地址范围叫作“区间 IP 地址范围”，将用于分配的 IP 地址范围叫作“范围 IP 地址”。

> [23] 如果我们做了冗余配置，则应将下一跳设为共享 IP 地址。

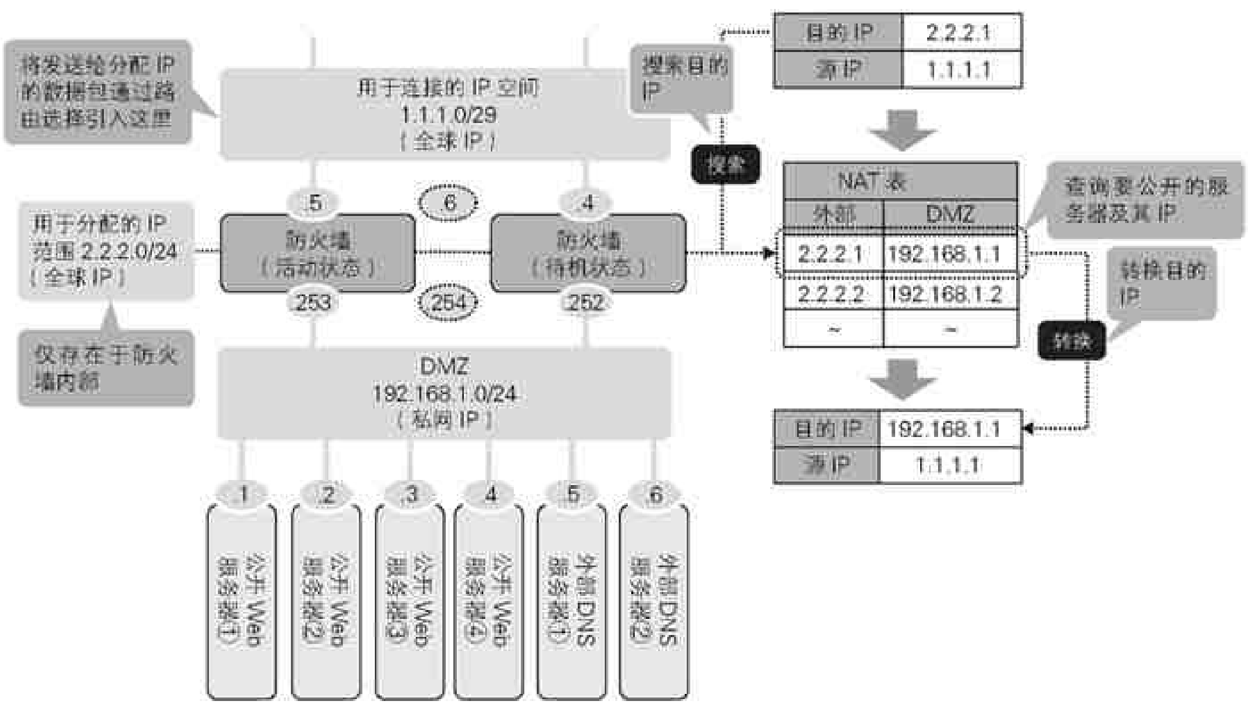


图 **2.3.20**　入站通信的 **NAT** 为一对一的形式

2.3.4.3　通过出站通信转换地址

出站（由内部去往外部的）通信是针对互联网的通信，需要转换源 IP 地址。针对互联网的通信大多是通过 NAPT 将多个私网 IP 地址转换为一个全球 IP 地址。其原理是，先找出应从哪个 VLAN 将信息发到互联网上，然后将该网络地址作为源 IP 地址进行转换。

将 NAT 定义为一对一形式的公开服务器，其出站通信的定义步骤和入展通信的刚好相反。

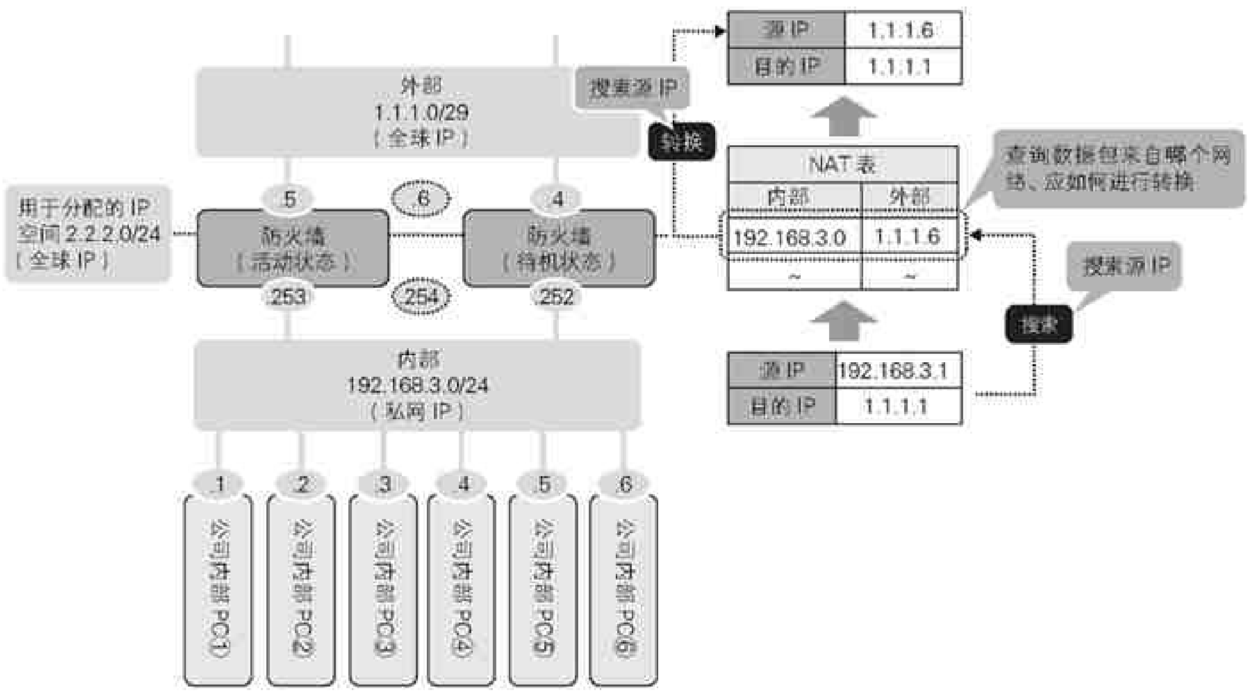


图 **2.3.21**　出站通信通过 **NAPT** 进行地址转换

第 3 章　数据安全设计和负载均衡设计

本章概要

本章将要说明服务器端从传输层到应用层的技术以及使用这些技术时的设计要点。

几乎所有我们使用或开发的应用程序都能够在网络中“流通”运行，因此信息流量也在持续地迅猛增加。对于这些激增的信息流量，服务器能够提供多大程度的安全保障，又能够完成多大数量的处理，这两点可以说是服务器端的关键所在。只有扎实地掌握技术规范，并且设计出最佳的信息安全环境和负载均衡环境，我们才能灵活应对越来越多的信息流量和日益复杂的应用程序要求。

3.1 传输层的技术

传输层用于高效地传输应用数据。网络层只负责将应用层的数据传输到目的节点，并不了解传输的是怎样的应用数据，也不关心该应用数据有着怎样的通信控制要求，而传输层恰恰能够填补网络层的这些缺点。它是网络层和应用层之间的桥梁，能够确保通信的灵活性。

3.1.1 通过端口号划分服务器进程

传输层是根据应用程序控制通信的层，网络层则是确保不相邻的节点（不在同一个网段中的节点）也能够相互连接的层。传输层除了能确保这种连接性以外，还能够添加传输报头，达到按应用程序要求控制通信的目的。在传输层执行的报头处理叫作分段，被分段的数据叫作报文段。传输层上定义了各种不同的分段方式。

传输层是 OSI 参考模型中自下向上数的第四层。如果是发送数据，首先要接收来自会话层的应用数据，然后将其分段并交给网络层；如果是接收数据，首先要接收来自网络层的数据包，执行一个和分段恰好相反的处理后再交给会话层。下面就详细解说一下这两种处理。

我们先来看发送数据。来自会话层的应用数据中仅包含用户输入的信息和输出输入画面的信息，并没有应用程序的相关信息。这样是无法知道该应用程序需要怎样的通信控制以及使用了哪项服务的。因此，为了解决这个问题，人们在传输层为数据添加传输报头信息并进行分段，然后再将其交给网络层，如图 3.1.1 所示。其中，会使用 TCP 或 UDP 协议来表示数据需要怎样的通信控制，使用端口号来表示数据使用了哪项服务。如果应用数据较大，就把数据按 MSS（Maximum Segment Size，最大报文段长度）切分分段。关于 MSS 的内容详见 3.1.1.3 节。

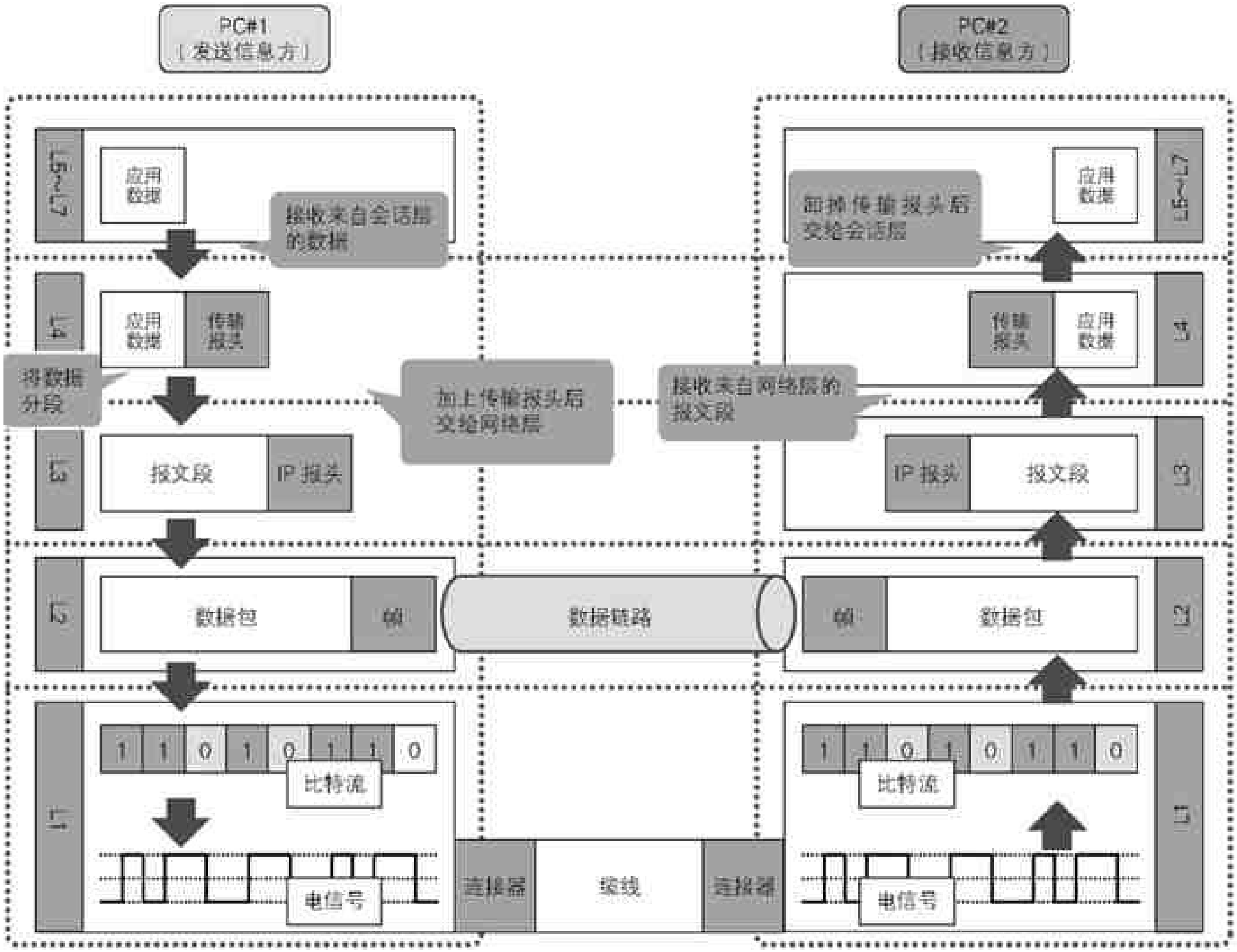


图 **3.1.1**　在传输层为数据添加传输报头

我们再来看接收数据的步骤。先是在网络层卸掉 IP 报头使数据变为报文段的形式，然后将报文段交给传输层。接下来在传输层卸掉报文段的传输报头使其还原为应用数据，最后交给会话层。这些处理步骤和发送数据时是恰好相反的。

3.1.1.1　传输层使用 **TCP** 和 **UDP** 两种协议

传输层是网络和应用程序之间的桥梁。应用程序对网络的要求是多种多样的，传输层将这些要求分为即时性（实时性）和可靠性两大类，分别通过不同的协议来实现。要求实时性的应用程序使用 UDP（User Datagram Protocol，用户数据报协议），要求可靠性的应用程序使用 TCP（Transmission Control Protocol，传输控制协议），使用哪一类协议在 IP 报头中的协议编号字段中定义。TCP 和 UDP 的差别见表 3.1.1。

表 **3.1.1　TCP** 和 **UDP** 的差别就是可靠性和实时性的差别

细分项目	**TCP**	**UDP**
协议编号	6	17
可靠性	高	低
处理负荷	大	小
即时性（实时性）	慢	快

用 **UDP** 进行快速传送

UDP 用于语音通话（VoIP，Voice over IP）、多目通信视频发布、名称解析、DHCP 等追求即时性的应用程序。它专注于传送本身，不具备可靠性，可省去繁琐的步骤，即时性非常高，其报文格式如图 3.1.2 所示。

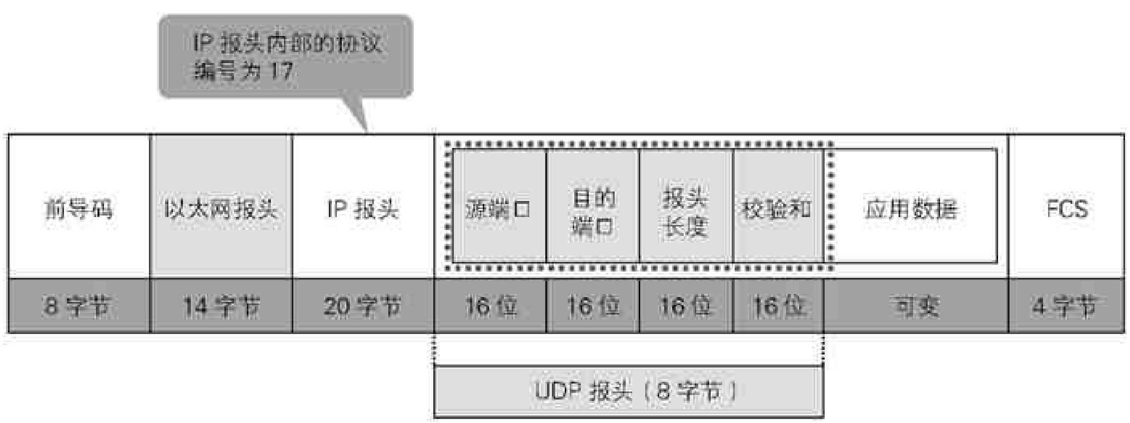


图 **3.1.2　UDP** 的报文格式

由于 UDP 追求即时性，封包格式也较为简单，只有 8 字节（64 位）。客户端用 UDP 对数据进行封包然后源源不断地传送，并不管服务器方面如何。服务器收到数据后利用 UDP 报头中所含的报头长和校验和去

检查数据是否正确，检验合格后才真正接收数据（图 3.1.3）。

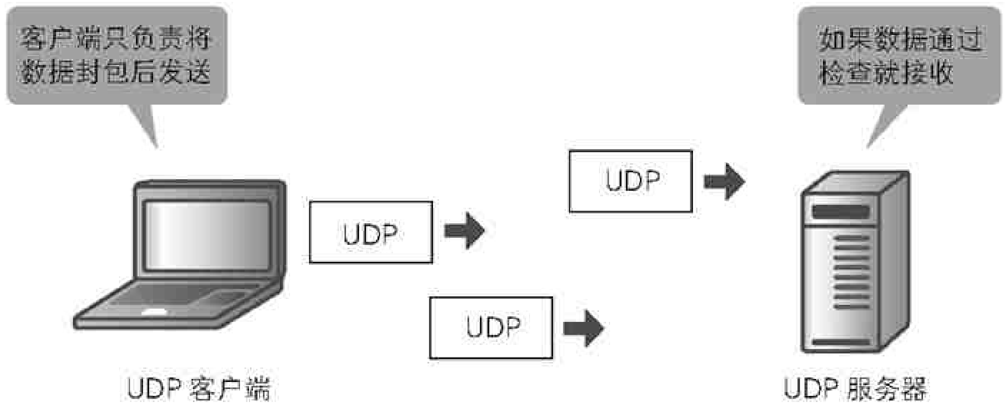


图 **3.1.3　UDP** 会源源不断地传送数据

用 **TCP** 进行可靠传送

TCP 用于邮件、文件的发送以及 Web 浏览器等追求可靠性的应用程序，其报文格式如图 3.1.4 所示。传送数据之前会先建立一个虚拟的连接通路，在其中进行数据交换。这个连接通路叫作“连通”。通过 TCP 进行数据交换的方法比较复杂，在 3.1.1.2 节中会详细说明。粗略地讲，就是每次传送数据时都要进行一个“我要发送了哦”“我收到了哦”的双向确认。由于每次都是一边确认一边传送，可靠性非常高。TCP 报头中包含着诸多信息，如图 3.1.3 所示。



图 **3.1.4　TCP** 的报文格式

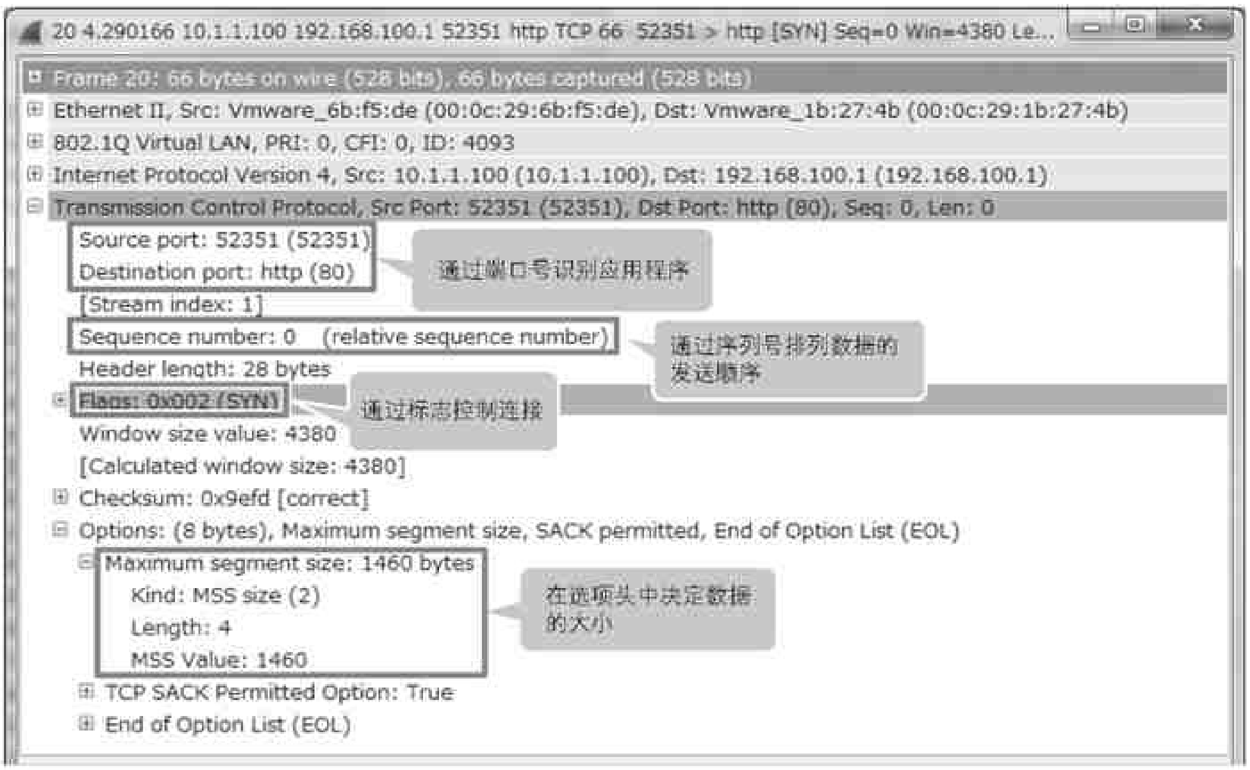


图 **3.1.5**　**TCP** 报头中包含着诸多信息

由于 TCP 追求可靠性，封包格式相对复杂，长达 20 字节（160 位）。而且它使用了大量报头对“我要发送了哦”和“我收到了哦”进行匹配确认，并高效地收发数据（图 3.1.6）。

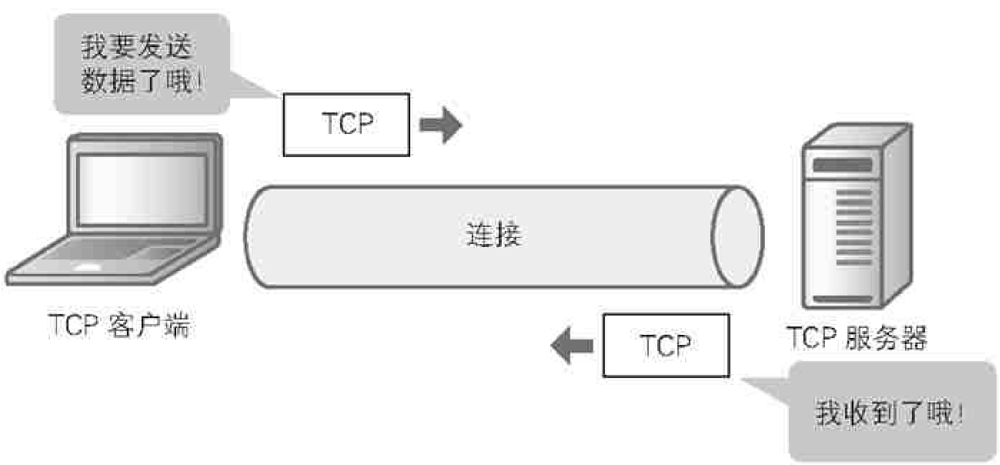


图 **3.1.6**　用 **TCP** 进行可靠传送

通过端口号识别应用程序

传输层还有一个很大的作用，那就是识别应用程序（服务）[1] 。前面提过，只要有网络层，数据就能够传送到目标节点。然而，接收数据的节点其实并不清楚应怎样处理该数据，这时就要用到传输层。传输层可以通过端口号识别出数据属于哪个应用程序。端口号和应用程序是匹配对应的关系，传输层认出目的端口号之后就能知道它属于哪一个应用程序。也就是说，传输层能够根据匹配对应的信息判断出应将数据交给哪一个应用程序。

> [1] 严格地说并不是识别应用程序本身，而是识别使用该应用程序的服务进程。为了浅显易懂，本章将“应用程序”和“服务进程”视为同一事物。

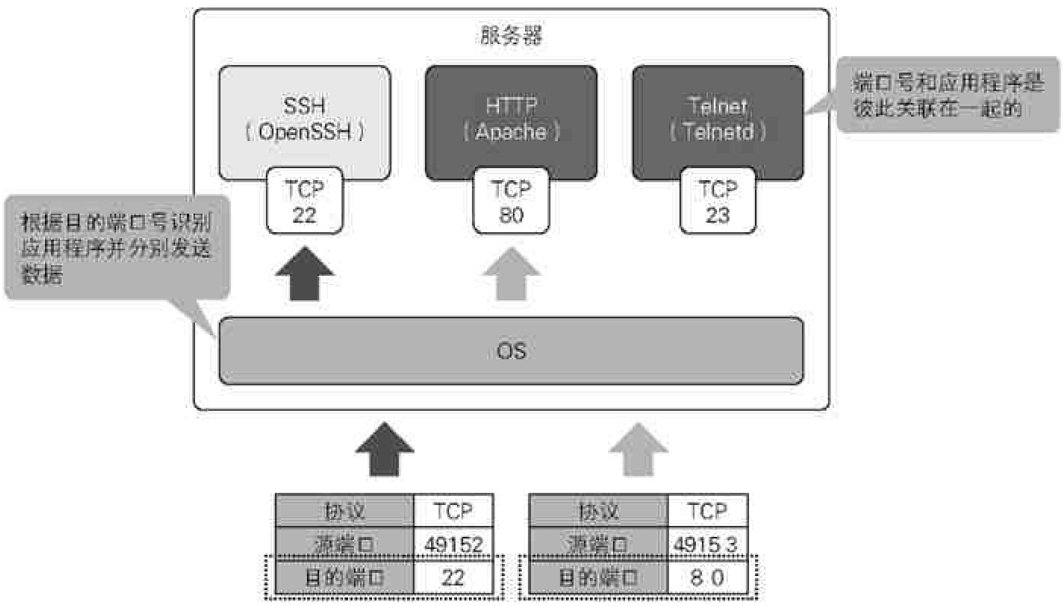


图 **3.1.7**　利用目的端口号识别数据所属的应用程序

端口号的数字范围在 0 ～ 65535（16 位）之间。根据进一步细分的数字范围和使用用途，人们将端口分为系统端口、用户端口、动态和 / 或私有端口这三种类型。

表 **3.1.2**　端口号分为三种

端口号范围	种类	解说
0～1023	系统端口（公认端口）	用于普通的应用程序

1024～49151	用户端口	用于生产商特有的应用程序
49152～65535	动态和/或私有端口	在客户端随机分配

- 系统端口

端口号为 0 ～ 1023 的端口是系统端口，更常用的叫法是“公认端口”。系统端口由 ICANN 中的 IANA（Internet Assigned Numbers Authority）部门管理，一般说来和服务器应用程序是一一对应的关系。例如，TCP/80 是用于 Web 浏览器的 HTTP，在 HTTP 服务器中使用；UDP/123 是用于时间同步的 NTP（Network Time Protocol），在 NTP 服务器中使用。典型的系统端口见表 3.1.3。

表 **3.1.3**　系统端口用于普通的服务器应用程序

端口号	**TCP**	**UDP**
20	FTP（数据）	—
21	FTP（控制）	—
22	SSH	—
23	Telnet	—
25	SMTP	—
53	DNS（区域传送）	DNS（名称解析）
67	—	DHCP（服务器）
68	—	DHCP（客户端）
69	—	TFTP
80	HTTP	—
110	POP3	—
123	—	NTP
137	NetBIOS名称服务	NetBIOS名称服务
138	—	NetBIOS数据报服务
139	NetBIOS会话服务	—
443	HTTPS	—
445	直接宿主	—

587	提交端口	—

- 用户端口

端口号为 1024 ～ 49151 的端口是用户端口。与系统端口一样由 IANA 管理，与生产商开发的特有服务器应用程序是一一对应的关系。例如，TCP/3389 用于微软 Windows 的远程桌面，TCP/3306 用于 MySQL 的数据库连接。典型的用户端口见表 3.1.4。

表 **3.1.4**　用户端口用于特有的应用程序

端口号	**TCP**	**UDP**
1433	Microsoft SQL Server	—
1521	Oracle SQL Net Listener	—
1985	—	Cisci HSRP
3306	MySQL Database System	—
3389	Microsoft Remote Desktop Protocol	—
8080	Apache Tomcat	—
10050	Zabbix-Agent	Zabbix-Agent
10051	Zabbix-Trapper	Zabbix-Trapper

- 动态和 / 或私有端口

端口号为 49152 ～ 65535 的端口是动态和 / 或私有端口，不受 IANA 管理，客户端在生成报文段时会随机生成源端口号。将上述范围内的端口号随机分配给源端口，客户端就能知道应将数据返回给哪一个应用进程。随机分配的端口号因 OS 而异，例如，Windows Vista/7/8 是 49152 ～ 65535，Linux 则默认为 32768 ～ 61000。Linux 使用的随机端口号范围在 IANA 指定的动态和 / 或私有端口的范围之外。

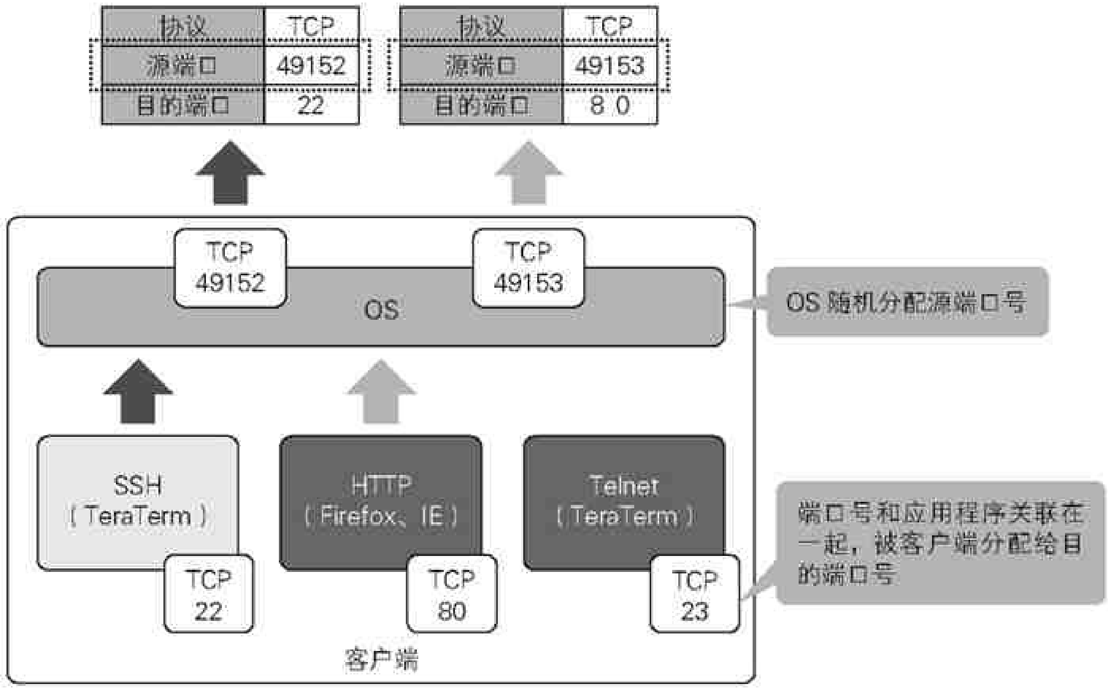


图 **3.1.8**　源端口号是由客户端随机分配的

下面的图例显示了接收和发送数据时端口号与应用程序的对应关系。

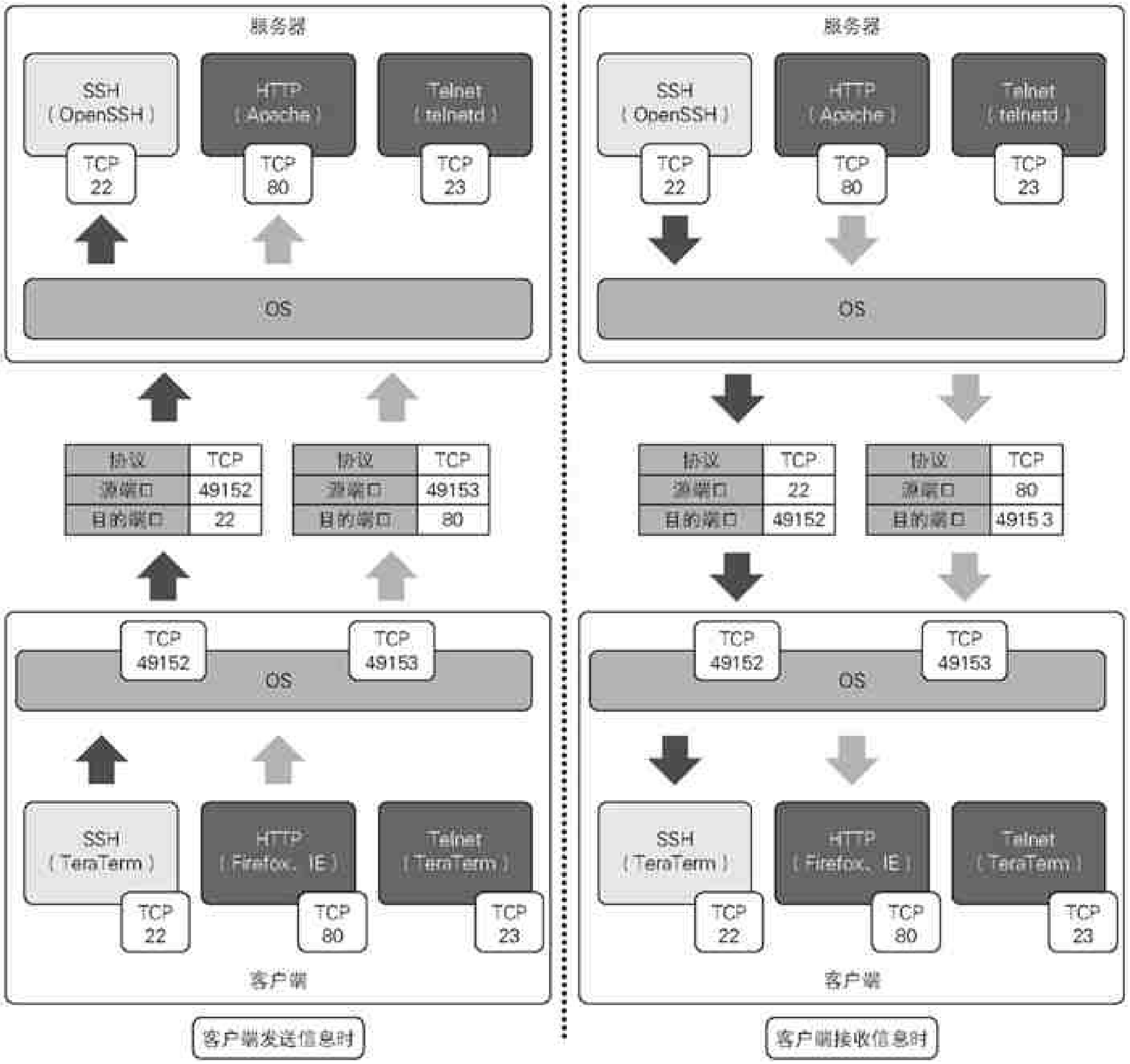


图 **3.1.9**　端口号是和应用程序关联在一起的

通过标志互相传达连接的状态

TCP 通过“控制位”管理连接的状态。控制位由长度为 6 位的标志构成，如表 3.1.5 所示，每个位都代表一定的含义。

关于这些 6 位标志的用法将在后面详细说明。

表 **3.1.5**　控制位由长度为 **6** 位的标志构成

控制位					
URG	**ACK**	**PSH**	**RST**	**SYN**	**FIN**

Urgent	acknowledgement	push	reset	Synchronize	finish
表示紧急的标志	表示确认响应的标志	将数据交给应用程序的标志	强制断开连接的标志	释放连接的标志	结束连接的标志

3.1.1.2 TCP 的工作原理比较复杂

为了保证连接的可靠性，TCP 执行的处理都比较复杂。本书将 TCP 的处理分成“建立连接时”“连接时”和“结束连接时”三个阶段，分别进行讲解。

TCP 始于三次握手

首先我们来看建立连接时。TCP 连接必须从三次握手开始，我们可以把三次握手理解为发送数据之前的打招呼。这里有三个要点，分别是标志、序列号和选项头。

- 标志

  三次握手的标志必须按照 SYN、SYN/ACK、ACK 的前后顺序依次变化才行。这一系列作业完成之后，客户端和服务器变为 ESTABLISHED（已建立）的状态，生成一个叫作 TCP 连接的虚拟通信路径。前面说过，三次握手在 TCP 中毕竟只相当于打招呼，所以实际上它并不执行应用数据的交换。

- 序列号

  正如其名，序列号是表示数据顺序的编号，客户端按照这些编号去排列数据。

  初始序列号用于发送应用数据，它是在三次握手中决定的。开始连接的节点随机选择一个序列号（x）并发送 SYN 包，对方节点收到 SYN 包后同样随机选择一个序列号（y）并回应 SYN/ACK 包。接下来最初的节点在序列号中写入 x+1，在 ACK 号中写入 y+1，最后发送 ACK 包。x+1 和 y+1 就是这两个节点各自的初始序列号。

  三次握手中的序列号和 ACK 号仅仅只是为了确定初始序列号而存在的，它们和交换应用数据时要用到的序列号和 ACK 号有着微妙

的不同。

- 选项头

 选项头起着重要的作用，那就是决定后续需要交换的应用数据的大小。应用数据太大的话是无法一次性发送的，必须切分成一定的大小分别发送才行。切分区域的最大长度叫作 MSS（Maximum Segment Size，最大报文段长度），三次握手将 MSS 植入选项头中进行数据交换。当双方节点的 MSS 值不一样时，取小的那一方使用。

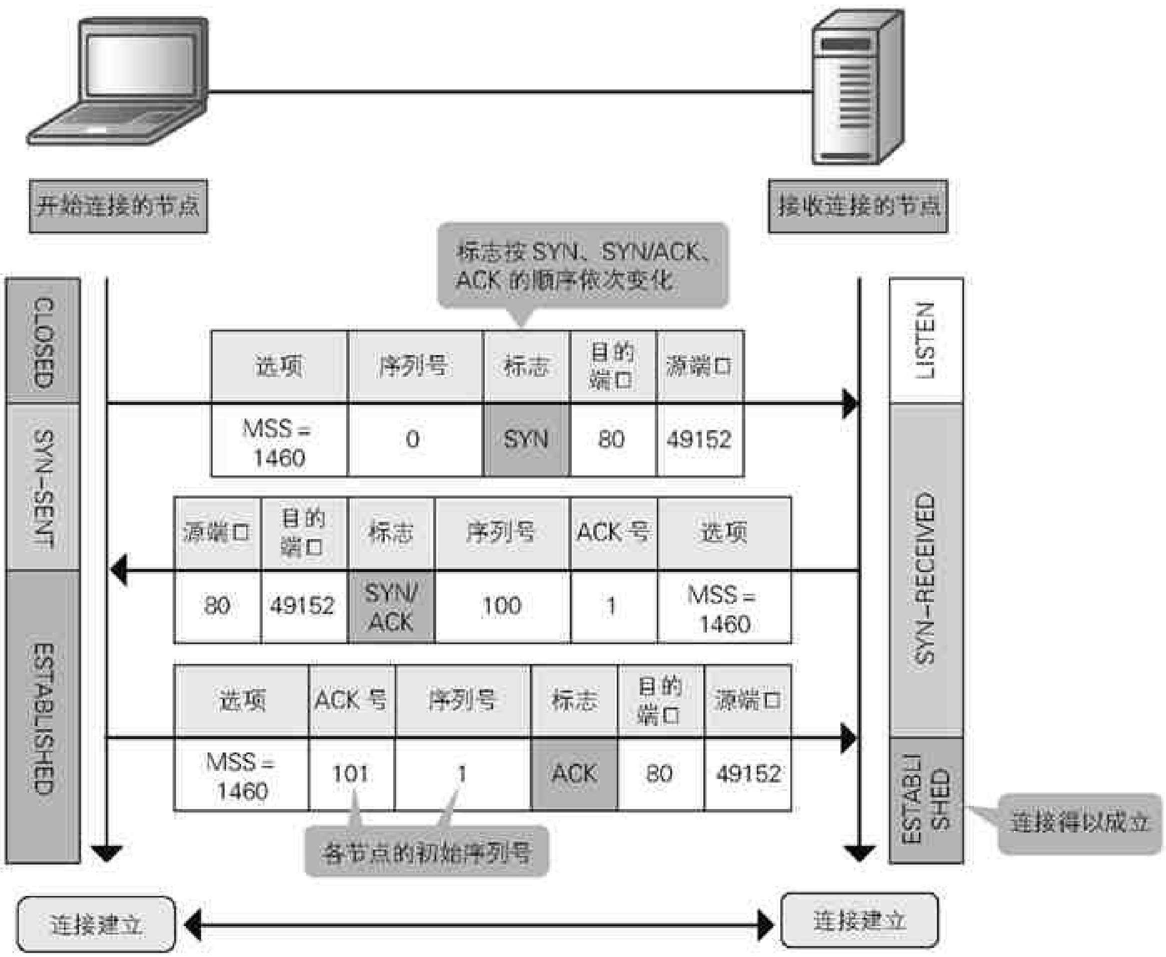


图 **3.1.10**　通过三次握手建立连接

一次性地高效发送数据

通过三次握手建立起连接之后，就可以发送应用数据了。为了能既保持

数据的可靠性又追求数据发送的速度，TCP 中有着很多的传输控制机制，本书将介绍其中的三个核心部分，分别为确认机制、超时重传机制和流量控制机制。

- 确认机制

  TCP 通过序列号和 ACK 号相互配合运作来保证数据的可靠性，这项机制叫作确认机制。

  序列号是表示数据顺序的编号，TCP 必须按照顺序对数据进行发送和接收，因此我们可以根据序列号去判断当前已经发送了多少数据。初始序列号是在三次握手时随机分配的，用当前的序列号减去初始值得到的数字就能告诉我们已经发出了多长的数据。顺便提一句，数据以字节为长度单位，序列号的 32 位（2^{32} 字节，即 4 吉字节）用完之后即归零。

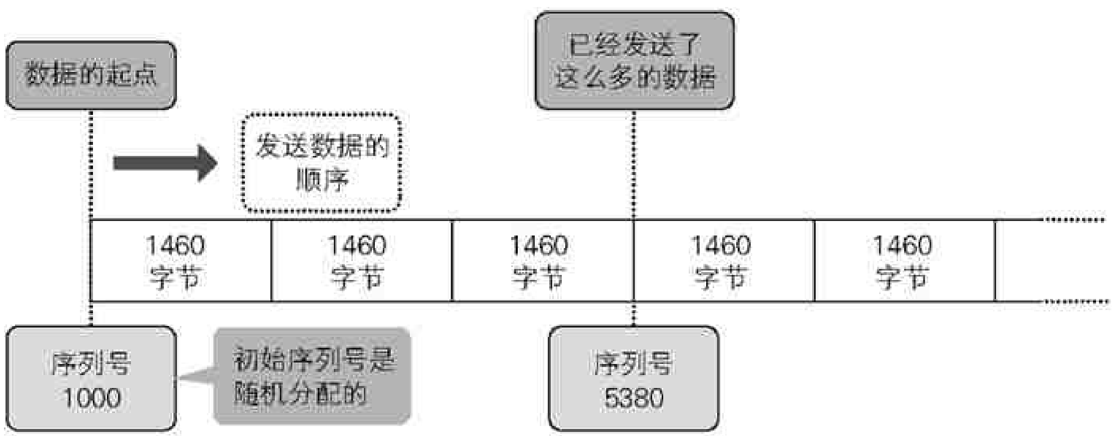


  图 **3.1.11**　序列号表示数据的顺序

  ACK 号表示接收信息的节点当前已经收到了多少数据，它会返回一个“序列号 + 数据长度”的数值，相当于告诉对方“我已经收到了这么多（序列号 + 数据长度）的数据哦，把下一个发给我吧！”ACK 号就是即将发送的数据的序列号。

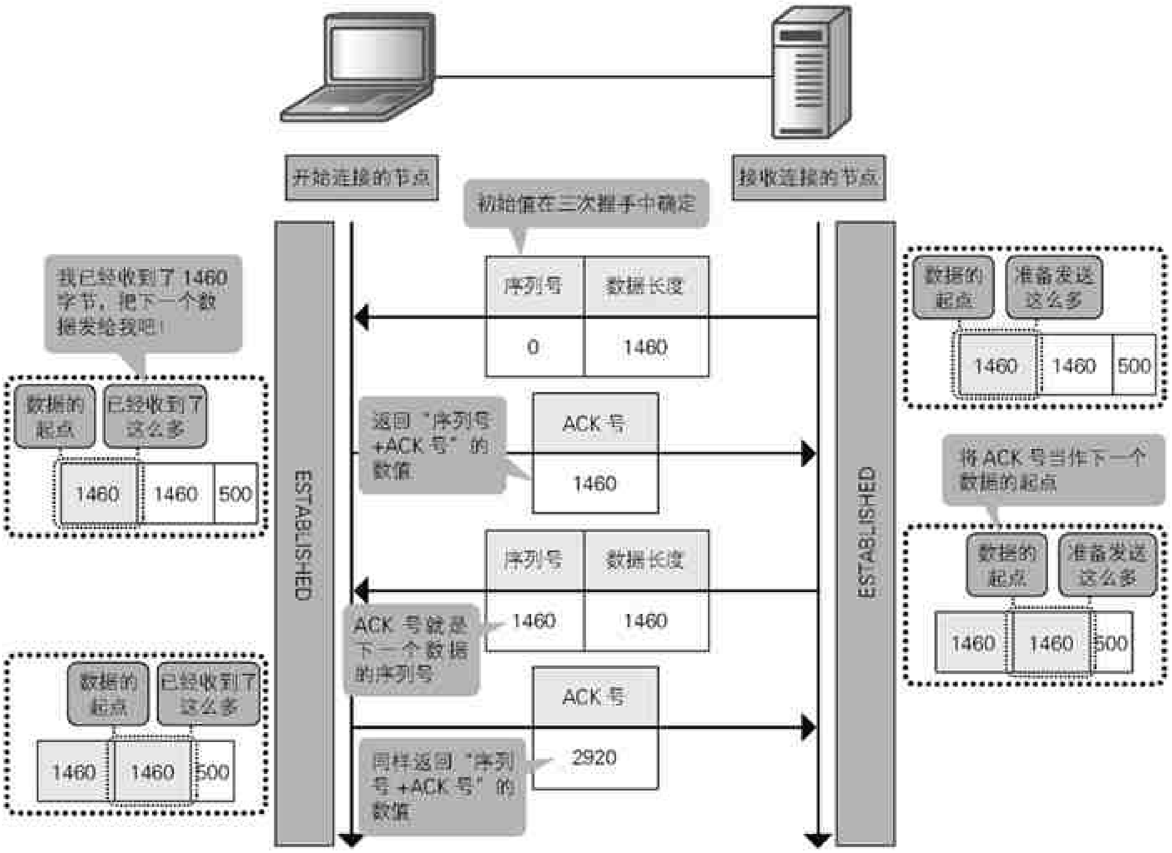


图 **3.1.12　ACK** 号告诉我们当前接收节点已经接收了多少数据

- 超时重传机制

确认机制让数据能够按照顺序排列，但是，数据并不总是能够按照顺序收发或排列的。比如，当设备发生故障时报文段可能会意外地消失；作为优先级控制（QoS）的结果，报文段也可能会被刻意丢弃。这时候我们就要用到 TCP 中请求重新传送丢失数据的功能，这项功能就叫作超时重传。

超时重传也要用到序列号和 ACK 号。假设服务器发送了第二个数据（序列号：x），但该数据不知在何处丢失了，这样客户端就不会返回确认响应的 ACK 包（ACK 号：x）。遇到这种情况，服务器在等待一定时间之后就会重新发送数据（序列号：x），这个等待的时间叫作 RTO（Retransmission Time Out，重传超时），RTO 是根据报文段的往返时间 RTT（Round Trip Time，往返延迟时间）自动计算出来的。

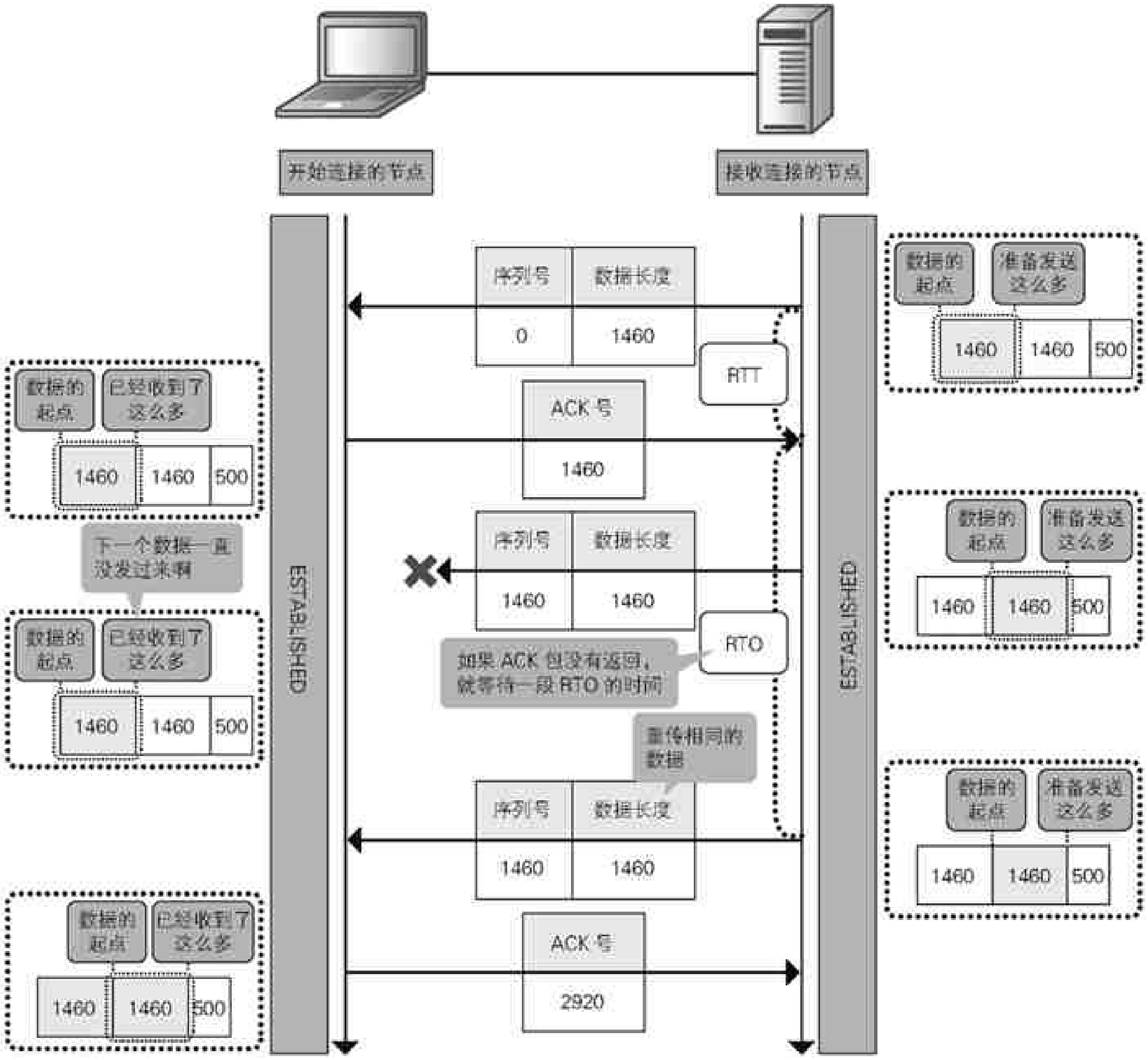


图 **3.1.13**　**ACK** 包没有返回时，服务器将重传数据

- 流量控制机制

确认机制和超时重传机制都属于提高通信可靠性的功能，不过在通信中，可靠性固然重要，速度也同等重要。因此 TCP 中还配备了流量控制功能，它能够在确保可靠性的同时提高传输效率。

流量控制是一种用于在连接上发送大量报文段的功能。如果每收到一个报文段就要返回一个确认响应，传输效率是无法提高的，所以流量控制批量接收报文段，收到后仅返回一个 ACK 响应包，传输效率因此而提高。在流量控制中起着重要作用的头是“窗口大小”这个概念。收到的应用数据先是被暂时存入接收缓冲区，然后才被交

给应用程序。目的节点在返回 ACK 包的时候，将接收缓冲区的空余容量以“窗口大小”的概念通知对方，防止发来的数据太大而发生溢出。源节点得知缓冲区的空余容量之后，在批量发送报文段的时候就会将数据大小控制在窗口大小的范围之内。

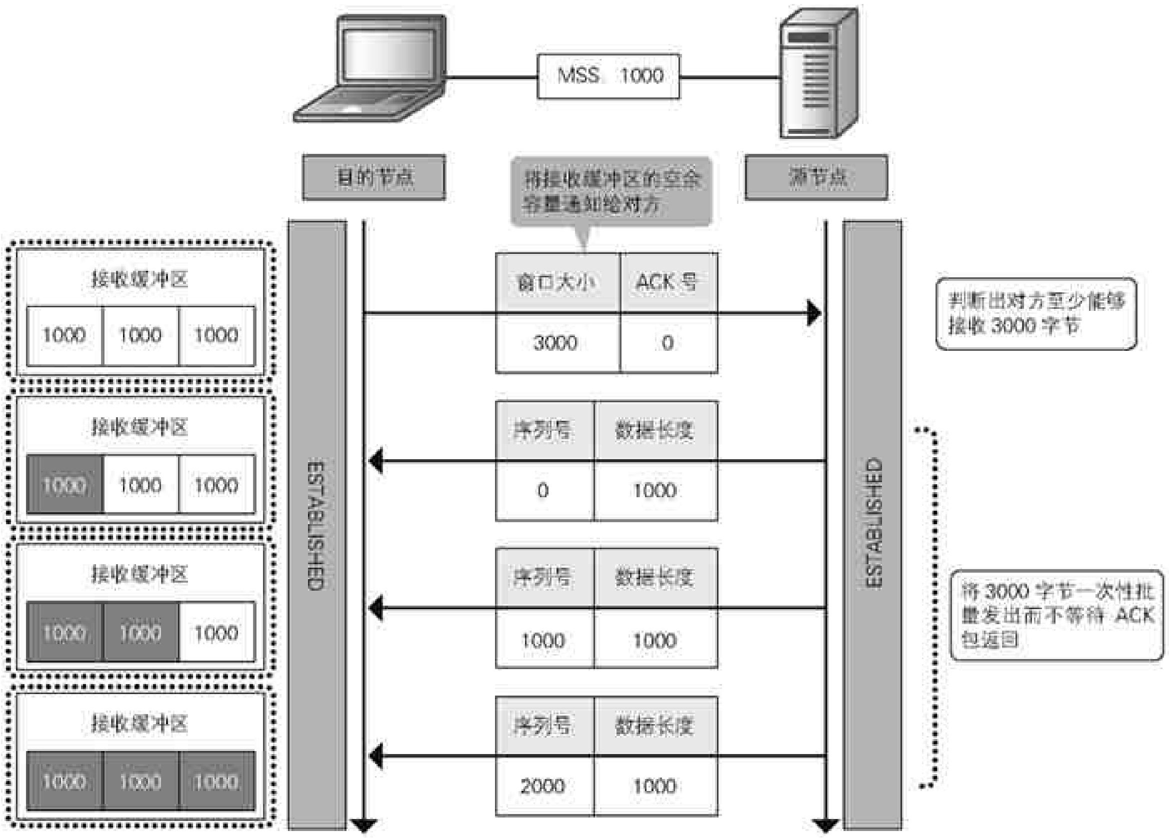


图 **3.1.14**　流量控制能提高传输效率

发送结束之后应完全关闭 **TCP** 连接

应用数据发送完毕之后应该完全关闭连接。如果未能完全关闭导致存在残余连接，将会无法建立新的连接。

这一步有两个关键要素，分别是 FIN 标志和 TIME-WAIT。

- **FIN** 标志

  结束连接的时候要用到 FIN 标志。FIN 标志表示“我再也没有要发送的数据了哦”的意思，由上层应用程序提供。FIN 可能由发送方和接收方的任何一方节点发出，发出之后，接下来的标志依次分别

为 FIN/ACK、ACK、FIN/ACK 和 ACK。请注意，这里要比开始连接时更加小心，不能弄错它们的顺序。

- **TIME-WAIT**

  前面讲过，建立连接的时候，依次经过 SYN、SYN/ACK、ACK 这几个标志之后连接就成立了。然而，关闭连接的时候，两端的节点却不是马上就关掉的。节点收到 FIN 标志后会等待一段 TIME-WAIT 的时间，超过该时间之后才关闭。这是为了避免重复使用同一序列号和端口号，相当于上了一道保险。

  在 RFC 中，TIME-WAIT 的推荐时长为最大报文段寿命（MSL，Maximum Segment Lifetime）的两倍。MSL 是 120 秒，因此 TIME-WAIT 的推荐时长为 240 秒。然而在实际环境中 240 秒稍微长了一点，可能会导致本地端口资源的枯竭，所以人们在设置时常常会刻意将其缩短一点。顺便提一句，TIME-WAIT 的默认时长因 OS 的种类和版本而异，Windows Server 2003/2008 为 120 秒，Linux OS 则为 60 秒。

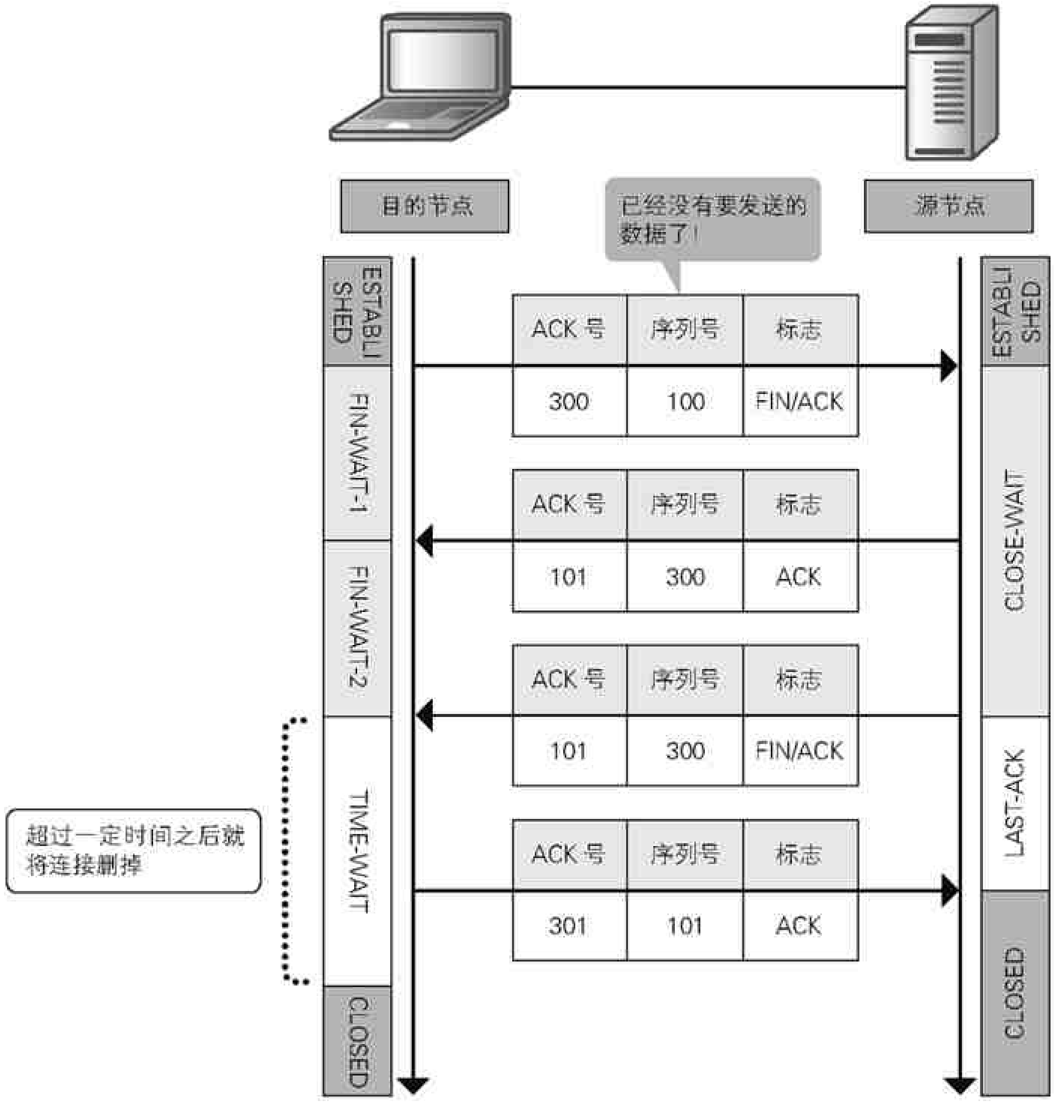


图 **3.1.15**　用 **FIN** 关闭连接

3.1.1.3　**MTU** 和 **MSS** 的差异在于对象层不同

MTU（Maximum Transmission Unit，最大传输单元）和 MSS（Maximum Segment Size，最大报文段长度）是两个支撑网络的重要元素，然而关于它们的介绍非常少，问题也很容易在它们这里滋生，下面我们就来理理这部分的头绪。

MTU 和 MSS 都是表示数据大小的用语，不过它们所指的对象层并不一样。MTU 指的是网络层中的数据大小。利用网络发送应用数据的时候，我们不能将太大的数据一次性地发出去，而应该对数据进行切分之后一点点地发出去。这时候用到的数据大小就是 MTU。不同的传输媒

介有着不同的 MTU 值，以太网默认的 MTU 为 1500 字节。

与 MTU 相对，MSS 指的是应用程序方面（由会话层到应用层）的数据大小。如果不对 MSS 进行特别的设置或者不使用客户端 VPN 软件，MSS 的值就是用根据“MTU－40 字节（TCP/IP 报头）”这个公式算出来的数字。例如，以太网的 MTU 值为 1500 字节，那么它的 MSS 值就是 1460 字节。

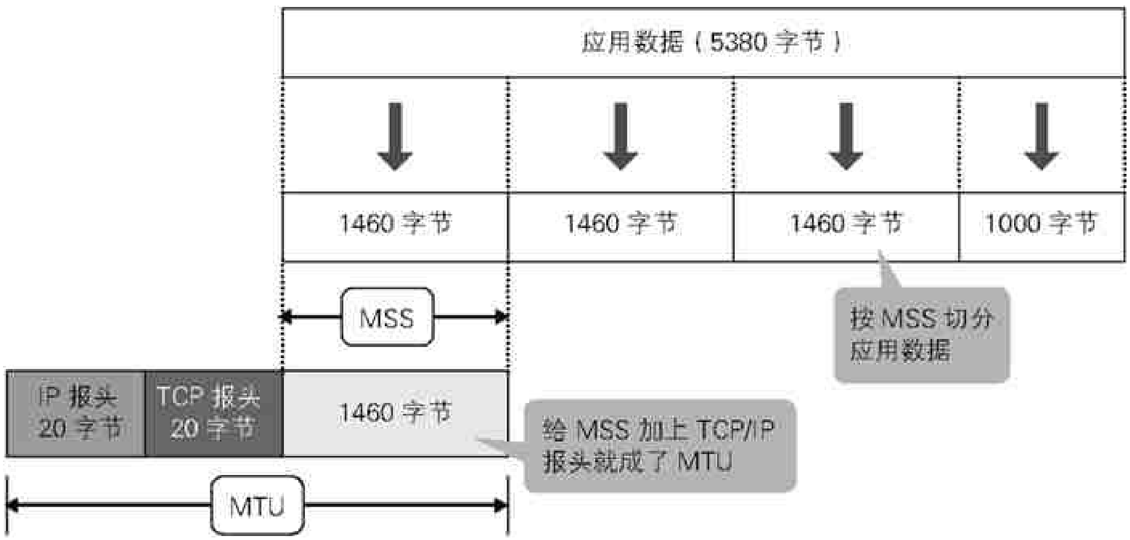


图 **3.1.16　MSS ＝ MTU － TCP/IP** 报头（**40** 字节）

有些环境中的某些路段 MTU 值偏小，在这样的环境中 MTU 和 MSS 很容易滋生问题 [2] 。我们知道网络无法发送大于 MTU 的数据包，于是，遇到某些路段 MTU 值偏小的环境时，为了让数据大小不超出 MTU 值的范围，我们必须在路由器或防火墙施加一定的处理才行。IP 报头的标志字段中 DF（Don't Fragment，禁止分片）位的值不同，处理方法也会有所不同。下面分别介绍一下 DF 为 0 和 DF 为 1 时的情况。

> [2] 当我们使用 NTT 东日本 / 西日本提供的“B FLET'S”或“FLET'S ADSL”通信服务时，PPP 之间、ISP 之间连接的系统开销会使 MTU 值变成 1454 字节。较之以太网的 MTU，FLET'S 部分的 MTU 值更小，这句话里面的“环境”就是指类似这种情况的环境。

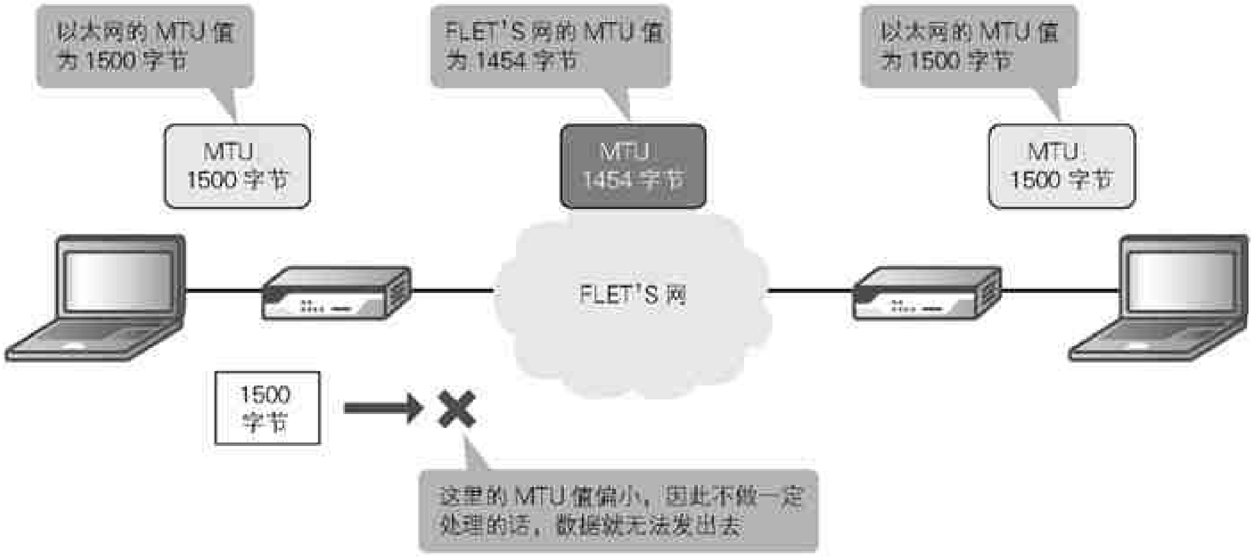


图 **3.1.17**　网络无法发送大于 **MTU** 的数据包

DF 为 **0**（允许分片）的数据包将被切分

DF 为 0 表示该数据包可以分片。既然可以分片，那我们就分片好了。这样，即使某个路段的 MTU 值比较小，也就不存在问题了，只要将数据包分片后发出去即可。如果一定要说有什么问题的话，只有一个，那就是分片处理的负荷问题。不过最近，路由器和防火墙的规格都相当高，执行分片处理绰绰有余，所以我们完全不必担心。

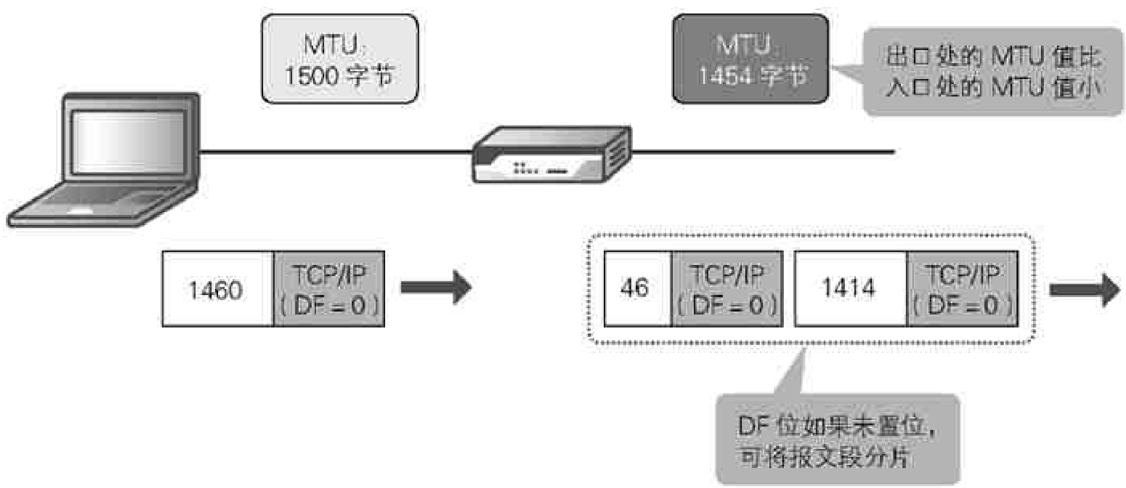


图 **3.1.18**　若 **DF** 为 **0** 则直接进行分片处理

DF 为 **1**（不允许分片）的数据包可有三种办法解决问题

DF 为 1 表示该数据包不可分片。既然不可分片，那我们就要想办法让数据包不分片也能够传送，或者执行一个使数据包能够分片的处理。这种情况下，有三种办法可供选择。

- 通过 **ICMP** 获悉 **MTU** 的值

ICMP 是一个不仅能用于通信确认，还在很多方面都默默无闻地做着贡献的协议，MTU 也会用到它。这部分按顺序解释可能会比较容易理解，我们来看下面的流程，ICMP 是按照这些步骤调整 MTU 的。

1 → 源节点将 MTU 值设为 1500 字节并发出数据包。

2 → 路由器将数据包入口处和出口处的 MTU 值进行比较，如果出口处的值较小，就通过 ICMP 将该值报告给源节点。顺便提一句，这时使用的 ICMP 类型为 3（Destination Unreachable，目的地无法到达），代码为 4（Fragmentation needed，需要分片）。

3 → 源节点收到 ICMP 后，用获悉的 MTU 值重新生成数据包并再次发送。

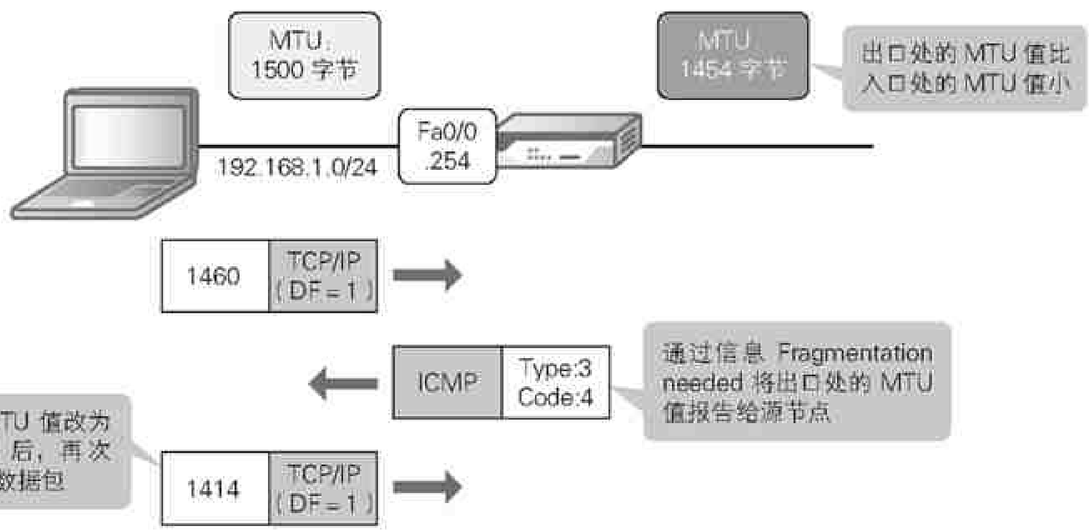


图 **3.1.19**　**ICMP** 将下一跳的 **MTU** 值报告给源节点

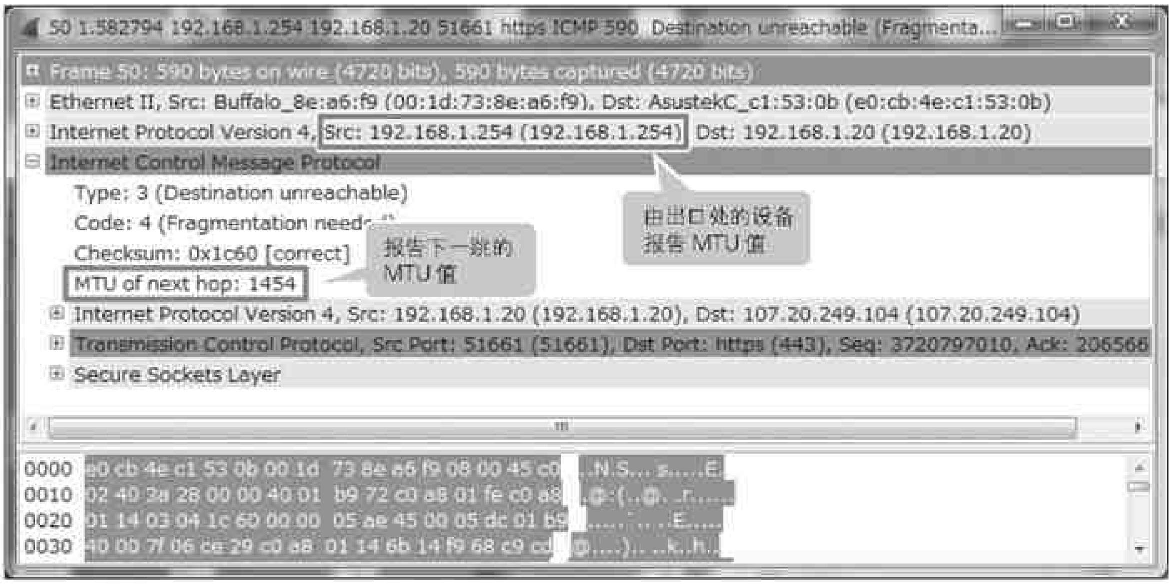


图 **3.1.20**　通过信息 **Fragmentation needed** 将下一跳的 **MTU** 值报告给源节点

通过 ICMP 解决分片问题虽然简单易懂，但是如果路段的某处拒绝 ICMP，就无法发挥作用了，我们务必要注意这一点。

- 改写 **DF** 位

接下来介绍改写（清零）DF 位的处理办法。既然无法分片，那么我们想办法让它能分片就是了。是否允许分片由 DF 位决定，所以我们可以将 DF 位清零使其允许分片，然后将数据包分片并发送出去。

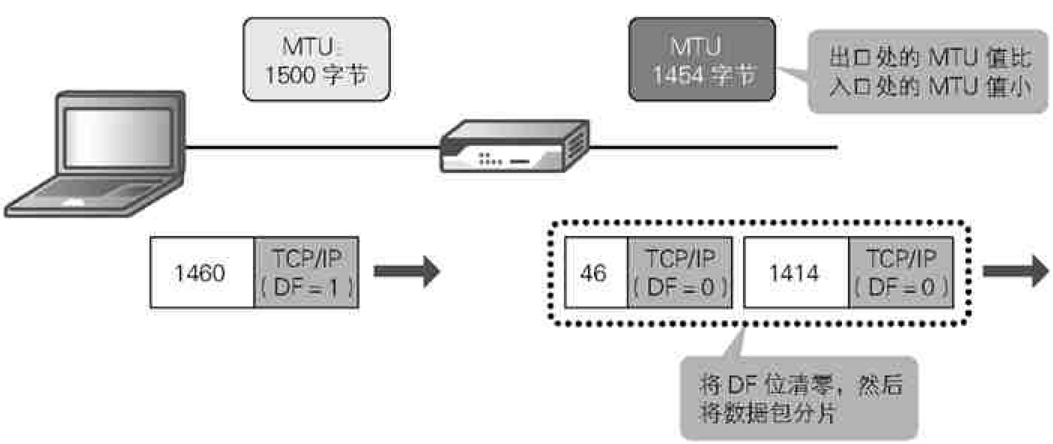


图 **3.1.21**　将 **DF** 位清零然后执行分片处理

- 在三次握手中将 **MSS** 值改小

  最后介绍调整 MSS 值的办法，这种办法也是最为常用的。我们知道 MTU ＝ MSS ＋ 40 字节，所以，为了让报文段的长度不至于超过 MTU 值，我们也可以将 MSS 的值改小，这样就连分片都不用了。MSS 由三次握手的 TCP 选项头决定，将该选项头中所含的 MSS 值改写成较小的数字，就能让报文段的长度保持在出口处接口的 MTU 值范围之内了。

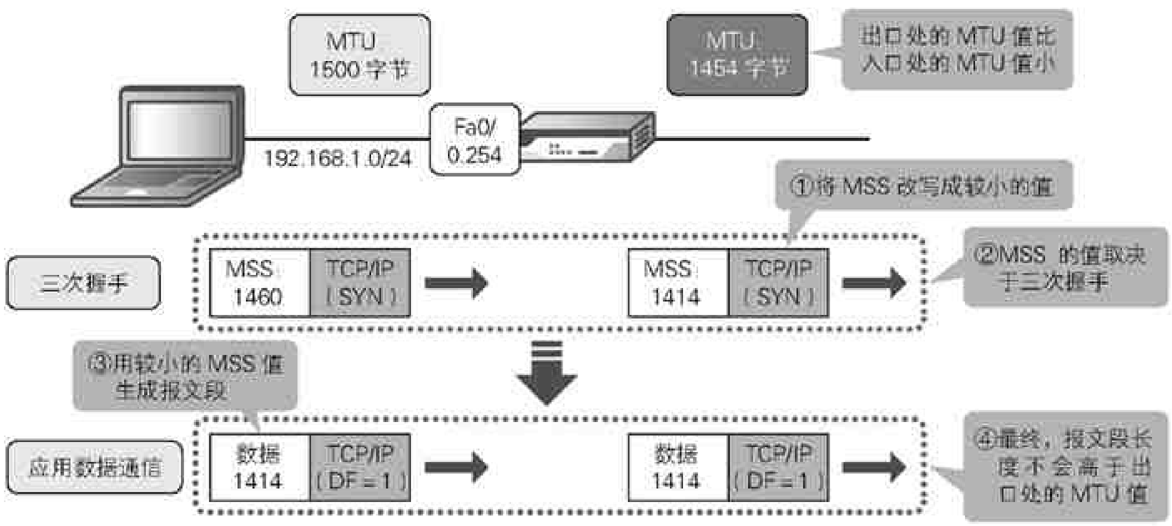


图 **3.1.22**　改写 **MSS** 的值

3.1.2　用防火墙守卫系统

在传输层运作的主要设备是防火墙。防火墙是一种基于网络层（IP 地址）和传输层（端口号）的信息来控制通信的设备，它遵照预先制定的规则对通信进行甄别，最终决定是放行还是拒绝通过，以此来守卫系统。

防火墙的通信控制功能叫作“状态检测”。状态检测是防火墙的一项核心技术，非常重要，我们务必好好掌握它的工作原理。

3.1.2.1　基于连接进行控制

状态检测是基于连接表的信息来执行的。连接表由“源 / 目的 IP 地址”“协议”“源 / 目的端口号”和“连接状态”等各种信息构成，在该表中可以看到都有哪些通信经过其中。

下面我们来看状态检测是如何通过连接表工作的。首先来理清一下网络环境和前提条件，这里我们假设客户端要在下图所示的环境中通过 HTTP 去访问 Web 服务器。

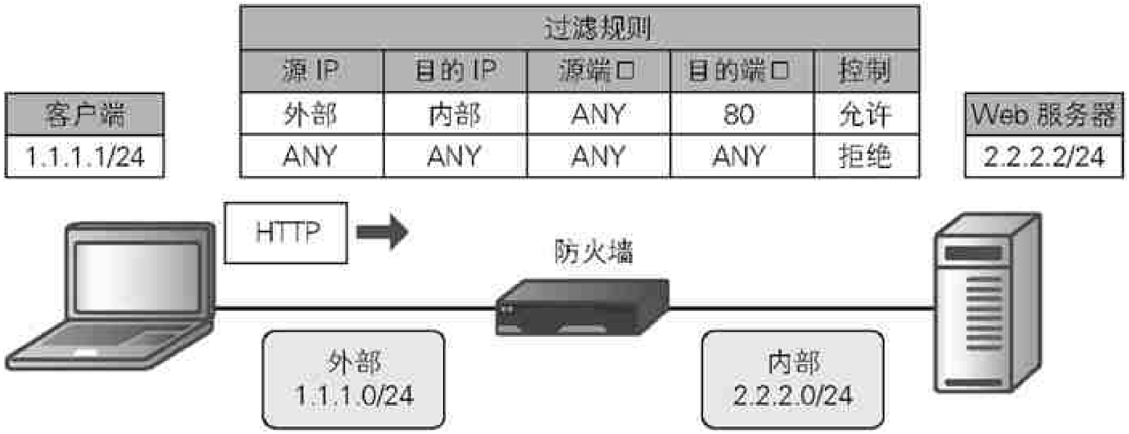


图 **3.1.23**　有助于我们了解状态检测的结构实例

1 →　防火墙通过外部接口收到 SYN 包后，将其和自身的过滤规则进行对照，发现与第一个条目相符，于是允许此通信通过。

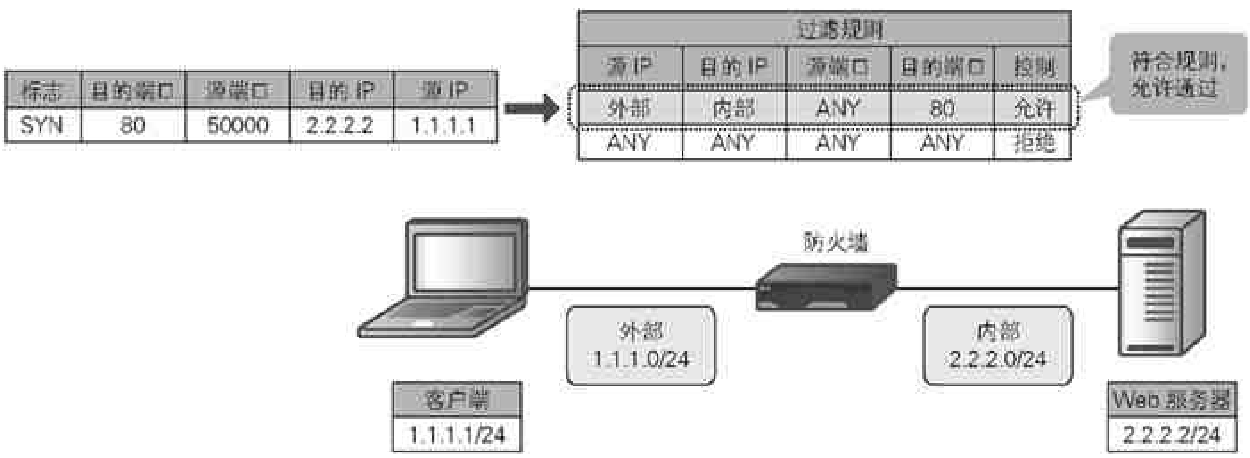


图 **3.1.24**　防火墙首先会查看过滤规则

2 →　允许通信通过之后，防火墙会在连接表中建立条目。与此同时，它还会动态添加一条规则，允许对应该条目的返回通信。然后，再将包转发给 Web 服务器。

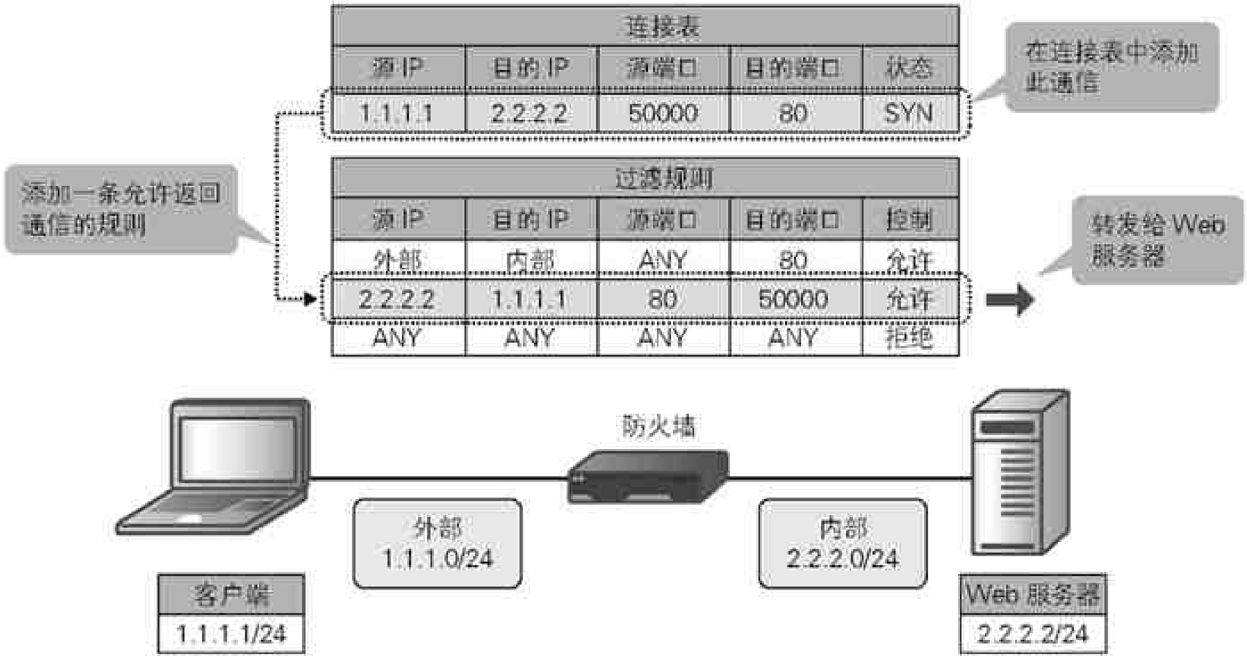


图 **3.1.25**　在连接表中建立条目，同时添加一条允许返回通信的规则

3 →　Web 服务器生成一个响应 SYN 包的 SYN/ACK 包，并将其返回给客户端。这条通信本来是会被拒绝的，但由于状态检测在 2 中添加了一条允许返回通信的规则，于是得以成功发送。

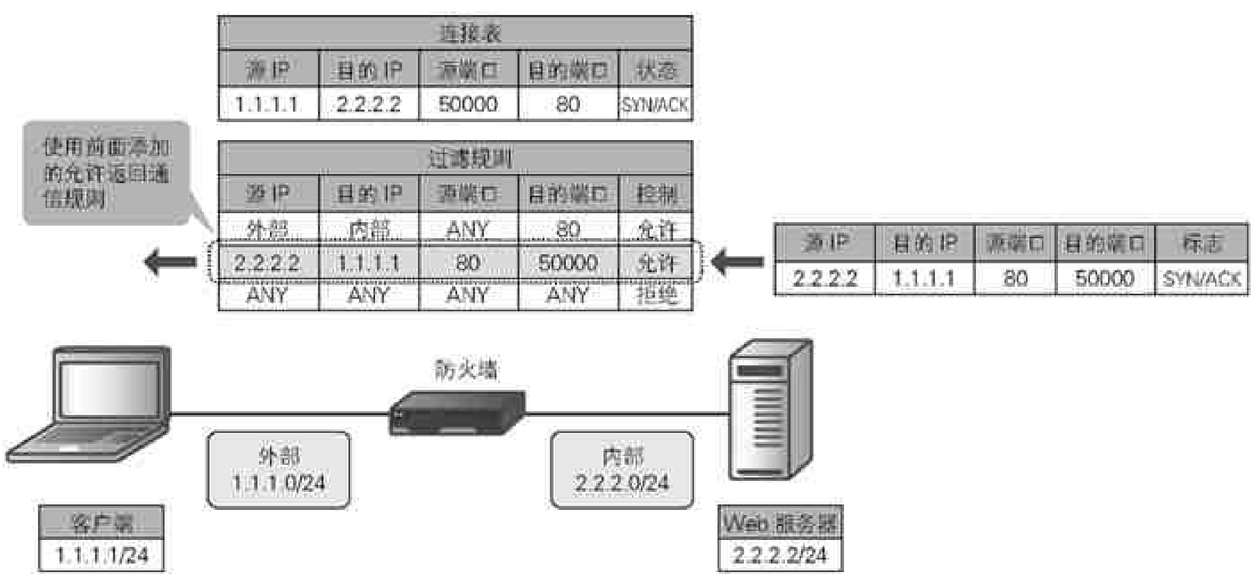


图 **3.1.26**　允许返回通信的规则生效，通信成功

4 →　客户端生成一个响应 SYN/ACK 包的 ACK 包，并将其返回给

Web 服务器。防火墙凭借该 ACK 包断定客户端和服务器之间的连接已建立，接下来在连接上开始交换应用数据。

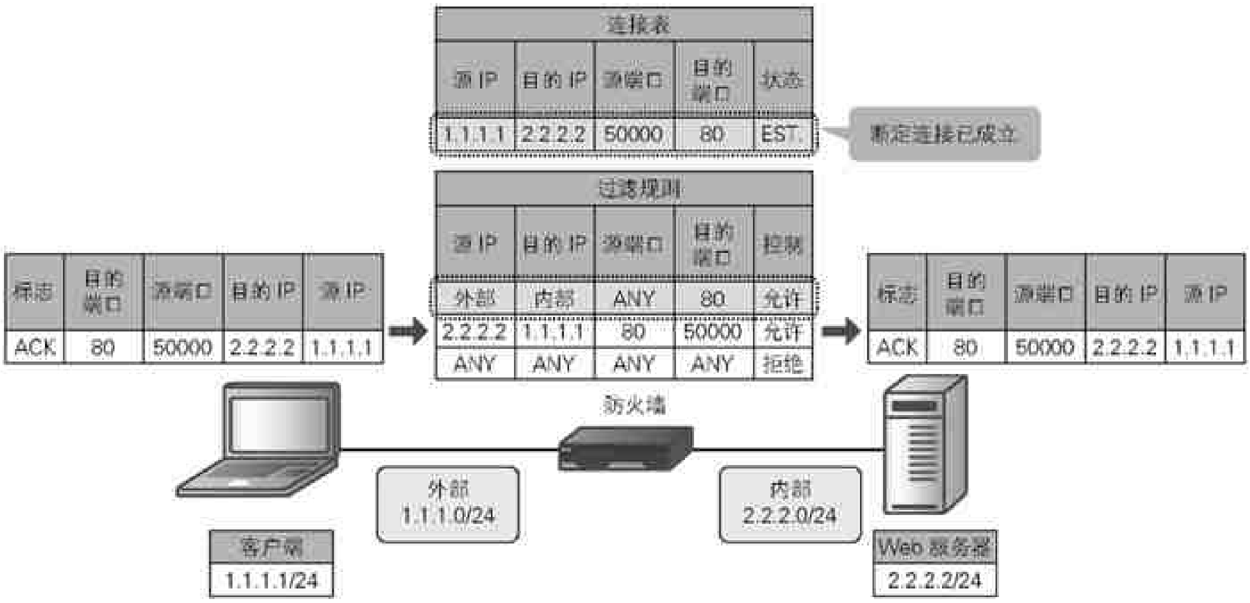


图 **3.1.27**　收到返回的 **ACK** 包后，防火墙断定连接已建立

5 →　连接建立之后，状态检测机制会去监控新建条目是否真的有用武之地。如果超过一定时间未被使用，该条目就会和允许返回的通信规则一起被强制性删除。条目的周期（条目自生成到被删除的时间）往往取决于使用的设备和 OS，有的是按应用程序设置的，有的则是按协议设置的，我们在设计之前一定要确认清楚。

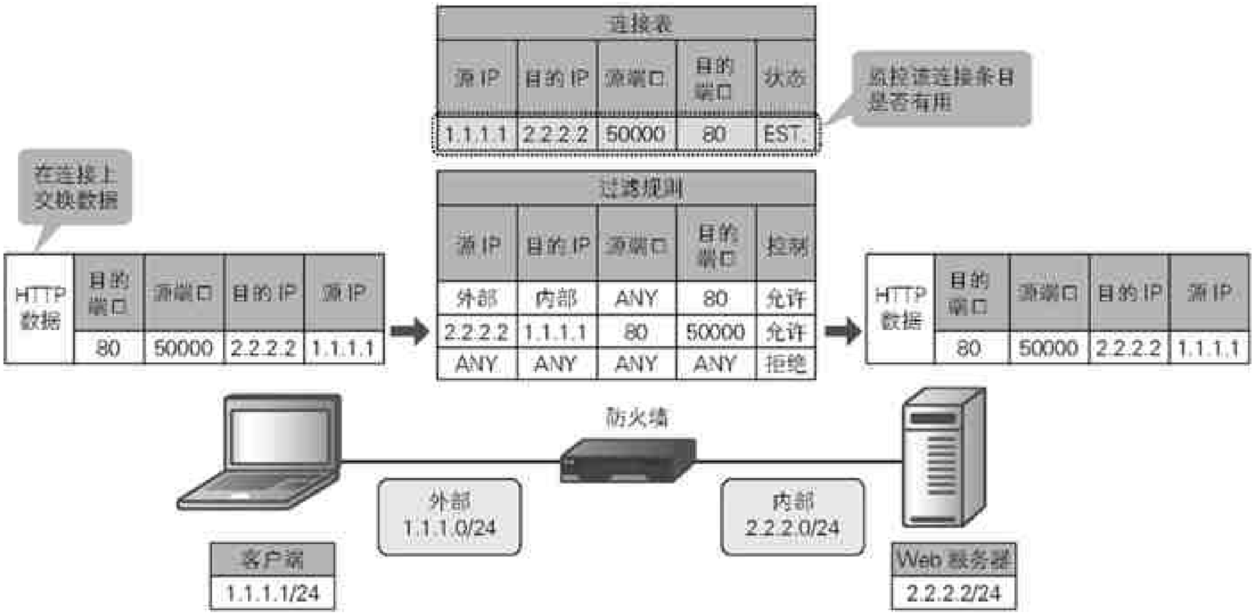


图 **3.1.28**　监控创建的连接是否有用

6 →　应用数据发送完毕之后，必须将连接完全关闭才行。防火墙在最后会根据客户端和服务器之间交换的报文段序列（依次为 FIN/ACK、ACK、FIN/ACK、ACK），将连接条目和允许返回的通信规则一起删掉。

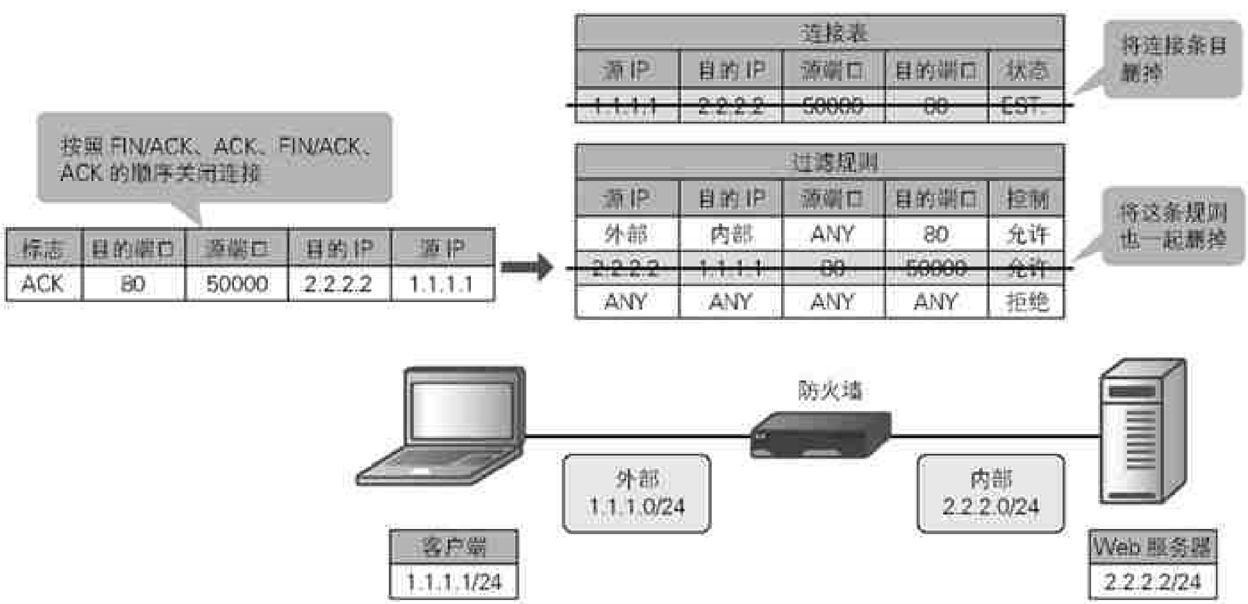


图 **3.1.29**　将连接条目和允许返回的通信规则一起删掉

3.1.2.2　状态检测和包过滤之间的区别

网络设备拥有的通信控制功能大致可以分为两种，一种是防火墙具备的状态检测功能，另一种则是路由器和 L3 交换机具备的包过滤功能。二者常常被混为一谈，但实际上它们还是有区别的，防火墙的通信控制功能不可能完全被路由器或 L3 交换机所替代。二者之间的最大差异在于是基于连接还是基于数据包。

前面已经介绍过，状态检测将通信视为一种连接并对其进行灵活的控制。实际上，除此之外它还会监控连接状态是否前后呼应，如果出现自相矛盾的情况就会断开连接。举个例子，如果没有收到 SYN 包就不应该有 ACK 包发过来，万一发过来了就表示出现了问题。状态检测对类似的非正常通信状态起着监控作用，一旦发现问题就会立刻将其阻断。

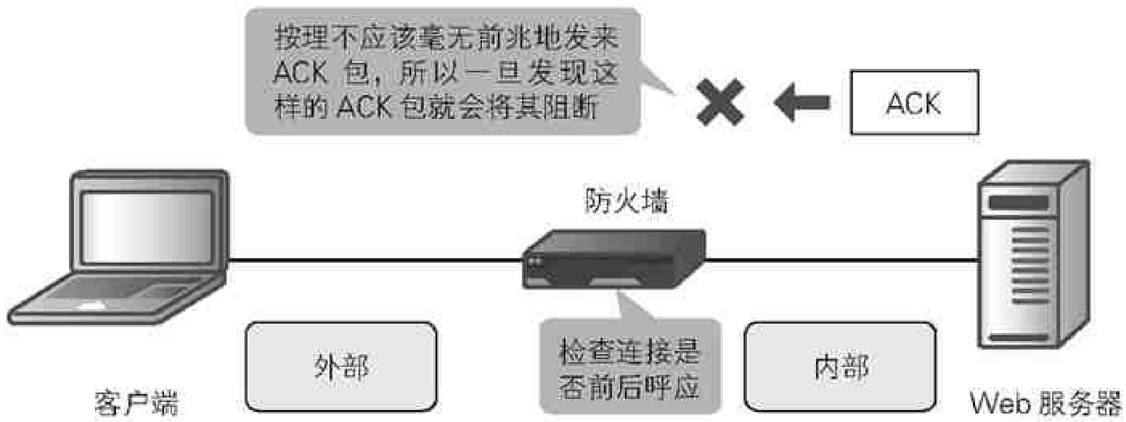


图 **3.1.30**　防火墙会检查连接状态是否前后呼应

包过滤则将通信视为一种数据包。每次收到数据包后，该功能都会将其和过滤规则进行对照，然后决定是允许还是拒绝该通信。到这里为止，包过滤的机制都是和状态检测一样的，它们的不同之处在于如何对待返回的通信。状态检测对返回通信采取的是动态等候的机制，包过滤则没有那么灵活，它需要将返回通信作为一个单独的返回通信，另外给出许可才行。

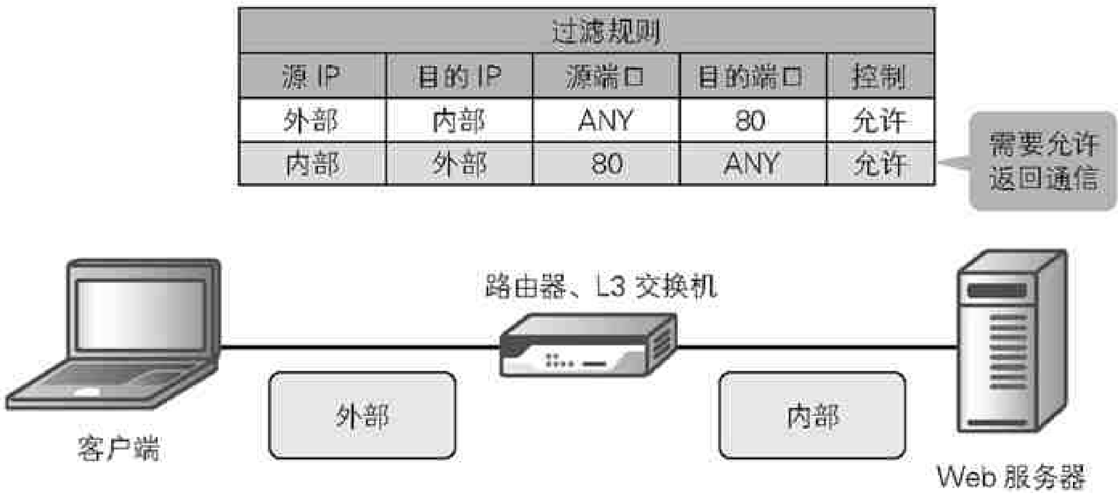


图 **3.1.31**　对于返回通信，包过滤必须另外给出许可

假设现在我们想要允许从某个客户端通往 Web 服务器的 HTTP 通信。使用状态检测时，只要有“允许从该客户端通往 Web 服务器的 HTTP”这一个规则，就能自动允许返回的通信；而使用包过滤时，除了允许从该客户端通往 Web 服务器的 HTTP 之外，还必须允许从 Web 服务器通往客户端的返回通信才行。而且，允许返回通信的规则还具有致命的易受攻击性，因为在从 Web 服务器通往客户端的通信中，源端口为 TCP/80，目的端口为 OS 随机选出的 TCP/1024～65535，这使得包过滤必须对大范围的目的端口号都给出许可。这在数据安全上存在着重大的问题，万一服务器被劫持，后果将不堪设想。

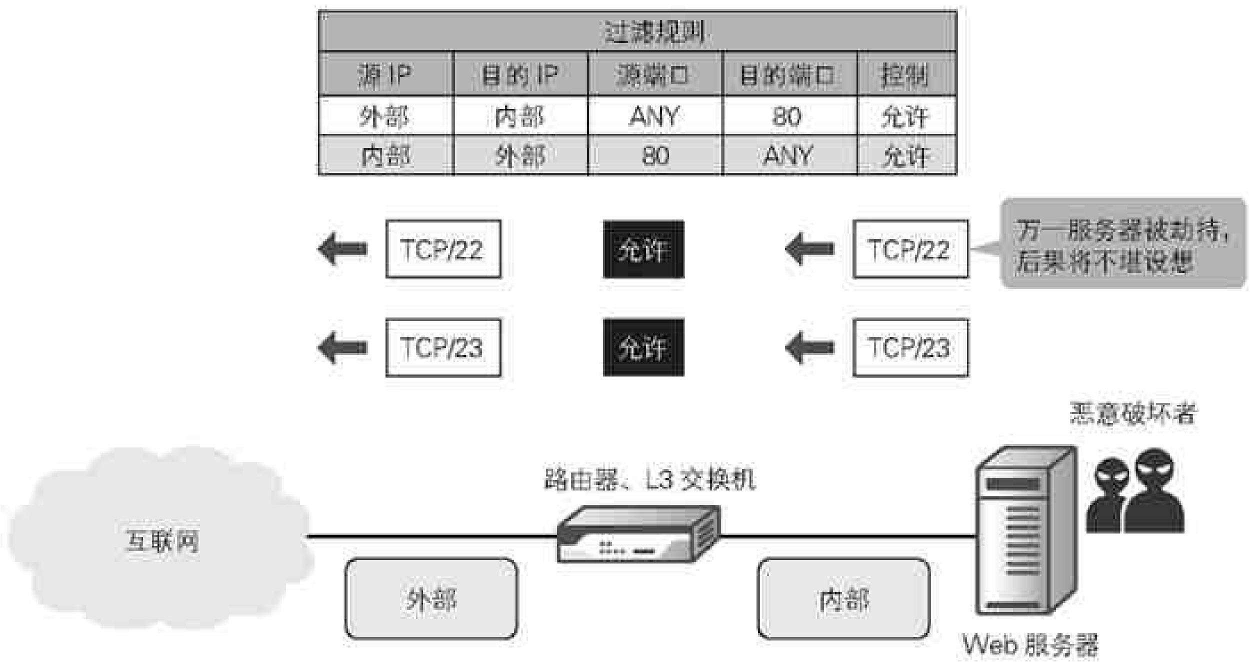


图 **3.1.32**　返回通信本身非常容易受到恶意攻击

有些设备可能会有一个 Established 的可选项，允许人们将带有 ACK 或 FIN 标志的包视为返回通信发放许可。但是恶意破坏者只要想办法操纵了标志，就能钻过滤规则的空子，所以这个可选项在数据安全上同样存在着问题。

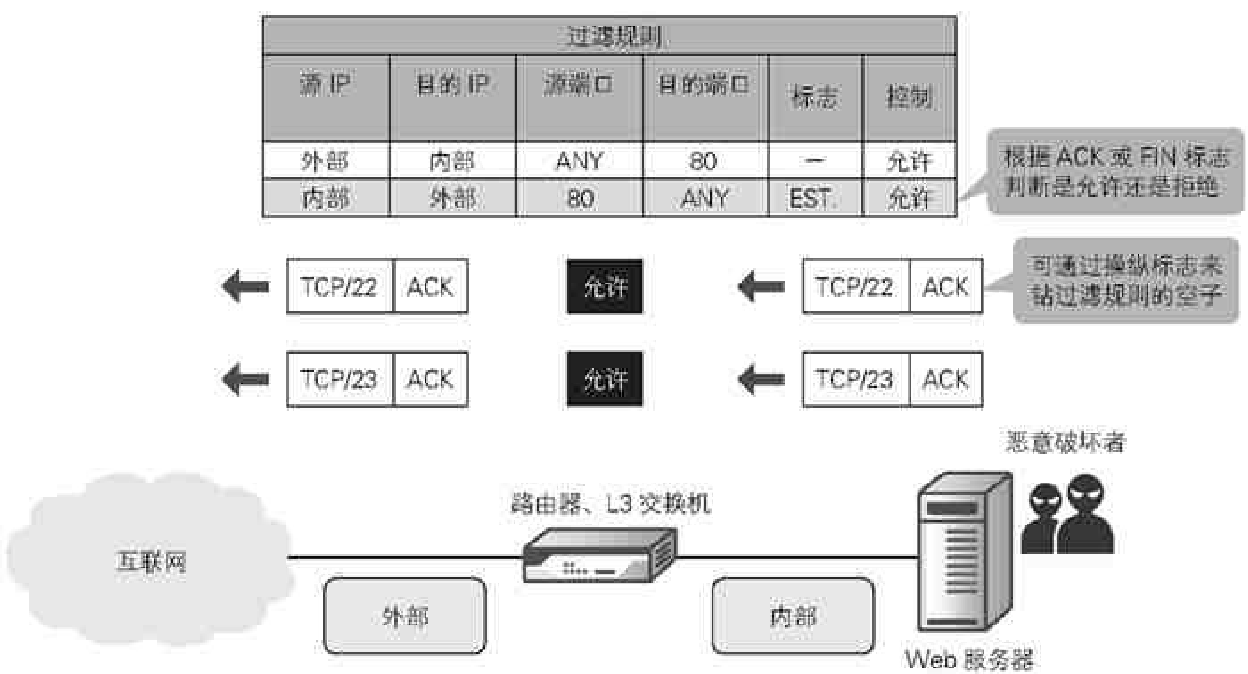


图 **3.1.33**　即便使用 **Established** 可选项，也无法完全控制通信

当然，出于对成本的考虑或者设备自身的原因，有时候我们也不得不使用包过滤功能，这时候一定要清醒地认识到在数据安全上它极易受到恶意攻击这一点。

3.1.2.3　防火墙在不断进步

防火墙是一种以现在进行时的节奏不断发展和进步的设备。过去有很多防御功能必须借助于专用设备才能实现，而如今的防火墙不仅能包揽这些功能，还能将防守范围扩大到应用程序的层面。面对日益复杂的网络安全威胁，防火墙越发能够灵活地应对，充分发挥它巨大的作用。

下面，本书将从两个方面来讲解防火墙技术的进步。一个是“功能的完善”，另一个是“防守范围的扩大”。

UTM 集多种功能于一身

其中一项进步叫作 UTM（Unified Threat Management，统一威胁管理）。简单地说，UTM 就是一种万能的设备，只要拥有一台 UTM，那么除了通信控制之外，VPN（Virtual Private Network，虚拟专用网络）、IDS（Intrusion Detection System，入侵检测系统）/IPS（Intrusion Prevention System，入侵防御系统）、反病毒、反垃圾邮件和内容过滤等功能就一应俱全了。以往人们必须借助专用设备才能实现各种不同的防御功能，现在却只要一台设备就足够了，于是设备成本和管理成本大大降低，UTM 也因此获得了迅速的普及。

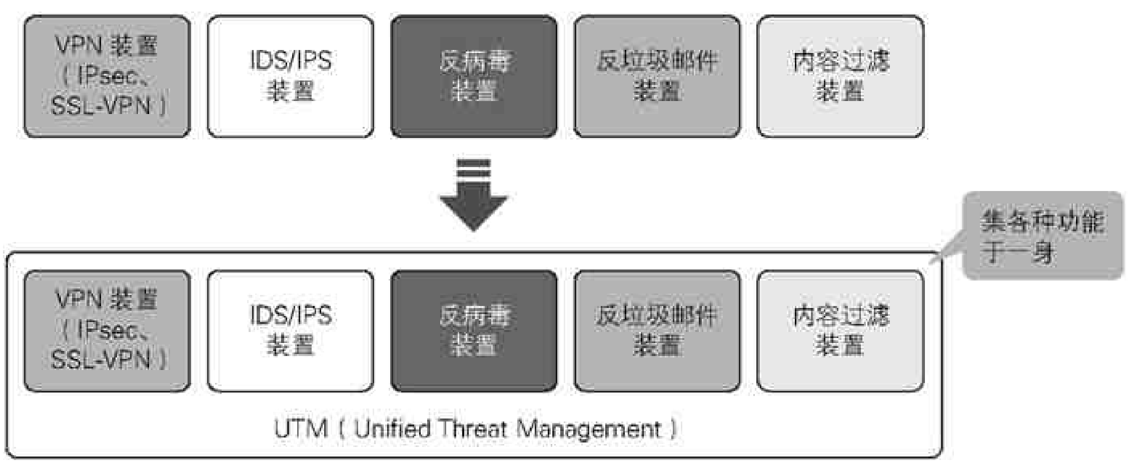


图 **3.1.34**　**UTM** 集多种功能于一身

那么，在这里我们就对 UTM 中的各种功能做一个大致的说明，只是扼要地介绍而已，具体使用的协议以及功能详情请参考相应的使用说明书。

- **VPN**

  VPN 是能够在互联网上建立加密的虚拟专线，连接诸多站点和用户的功能。它大致可分成两类，一类是点对点 VPN，另一类则是远程访问 VPN。

  点对点 VPN 是将诸多站点连接起来的一种 VPN。以往人们是用一对一的专用布线去连接站点的，这种方法的确比较安全也容易理解。然而随着距离的增加，也要增加支出的费用，而且费用还极其高昂，这个不利因素太大了，于是点对点 VPN 应运而生。点对点 VPN 在互联网上建立虚拟的专用线路，能像实体专线一样将站点连接起来，却只需要人们承担普通的互联网上网费而已，因此大大降低了成本。

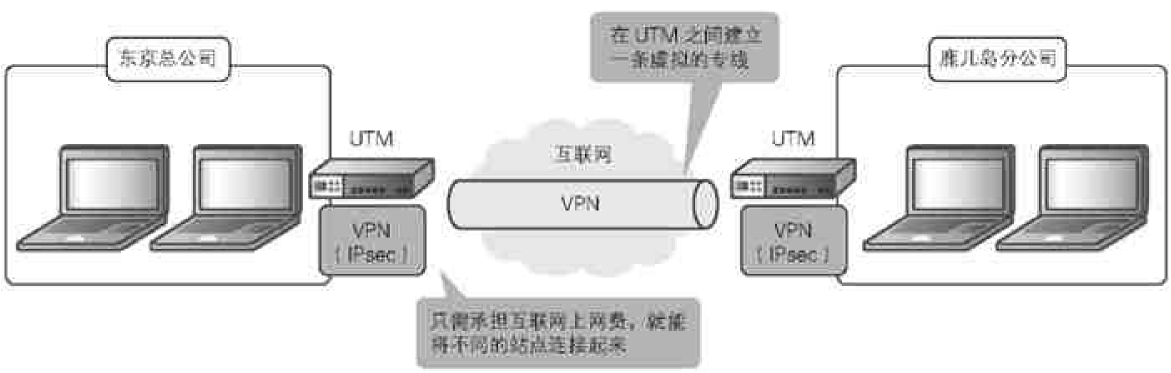


图 **3.1.35**　点对点 **VPN** 将站点连接起来

  远程访问 VPN 用于移动用户的远程访问。随着时代的进步，人们的工作方式也发生了变化。想象一下，员工因公外出时，在结束工作之后特意返回公司继续干活有意义吗？效率只会更低吧？这时候如果使用远程访问 VPN，通过 VPN 软件连接到公司网络，员工就能像自己在公司里一样正常工作了。远程访问 VPN 大致可分为 IPsec VPN 和 SSL-VPN 两种，以往 IPsec VPN 是主流，但由于和 NAPT 环境之间存在一定冲突而且无法用于有代理服务器的环境，

现在已经在逐步被 SSL-VPN 所取代。

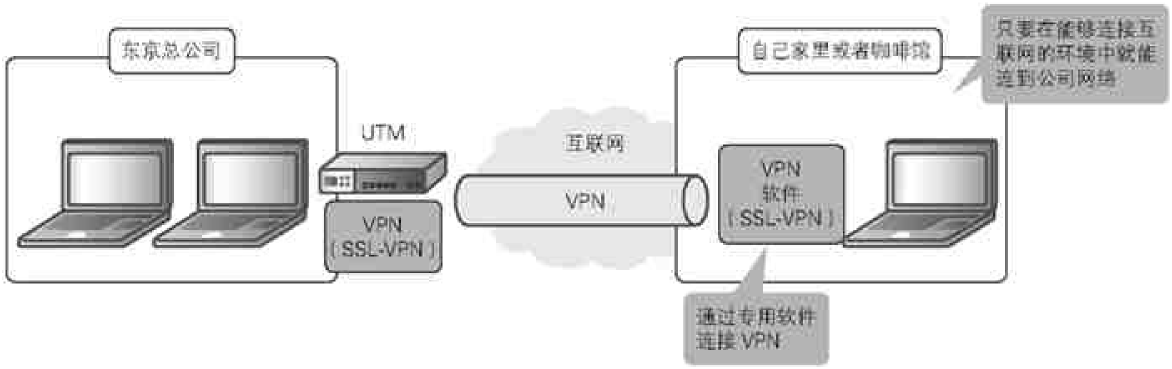


图 **3.1.36**　远程访问 **VPN** 可以从咖啡馆或自己家里连接到工作网络环境

- **IDS/IPS**

IDS/IPS 是通过观察通信情况检测出是否有入侵或攻击行为，或是对入侵和攻击进行防御的一种功能。IDS 只能检测，IPS 则兼具检测和防御的双重功能。

IDS/IPS 以一种叫作签名的形式保存着所有可疑的通信类型，签名相当于防病毒软件中的病毒码文件，通过自动或手动更新。IDS/IPS 将实际的通信和签名进行对照以检测和防御入侵。最近，网络攻击的手段越来越复杂，很难机械性地判断对象是否真的在企图入侵。日本大多是先用 IDS 检测一遍，然后视具体情况交给 IPS 去阻断攻击。IDS/IPS 的运行管理极其重要，我们不能实施之后就高枕无忧，而一定要根据实际环境情况对设置进行定制处理。

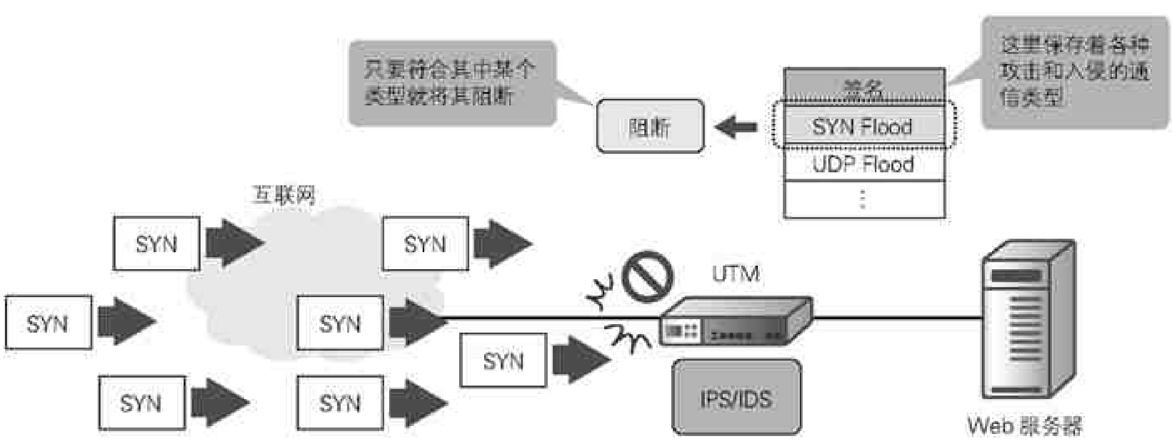


图 **3.1.37**　**IPS** 能阻断攻击

- 反病毒

反病毒是一种对抗病毒的功能，和 IDS/IPS 同样是基于签名运作的。它将收到的通信在 UTM 内部和签名进行对照并处理。签名可能是自动更新，也可能是手动更新。

在实施 UTM 时常常有管理人员问“有了这个我们就不需要防病毒软件了吧？”回答是否定的，因为 UTM 仅对经过它的通信进行监控。举个例子，当我们在自己家里或公用无线 LAN 环境中将已受到病毒感染的 PC 连到公司内部 LAN 的时候，UTM 是起不了任何作用的。数据安全必须采取多层防御的方式，这是一个基本原则，所以我们万万不可掉以轻心，要对接踵而至的网络威胁严阵以待才行。

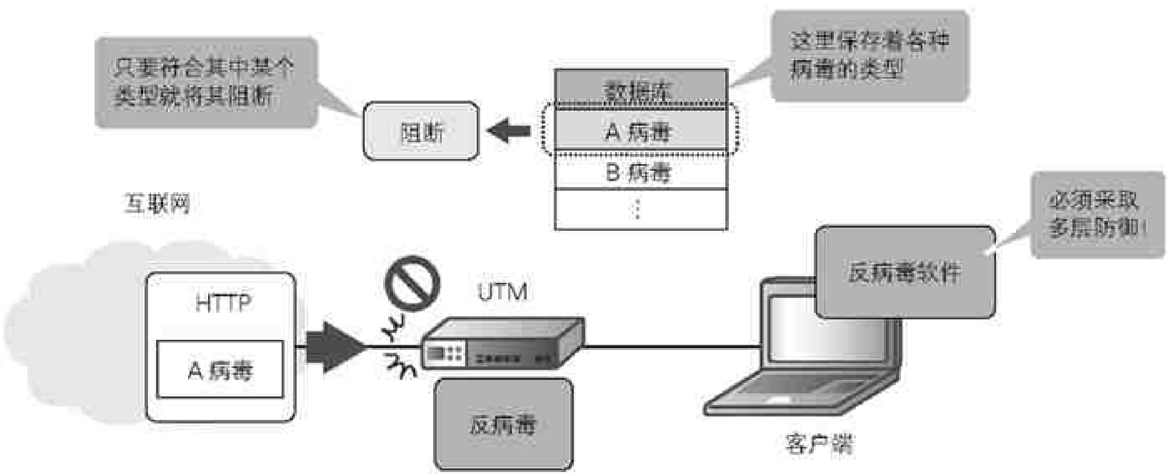


图 **3.1.38**　反病毒功能基于签名阻断病毒

- 反垃圾邮件

反垃圾邮件是一种对抗垃圾邮件的功能，是基于签名和信誉运作的。签名和反病毒、IDS/IPS 一样，是一个相当于病毒码文件的资料库。反垃圾邮件功能会根据邮件中所含的网址、图像、用语等各种要素去判断该邮件是否为垃圾邮件。信誉指的是根据邮件源 IP 地址去判断该邮件是否为垃圾邮件的一种技术，该技术会将发送垃圾邮件的邮件服务器的源 IP 地址保存到特定的数据库里，一有需要就对照该数据库来判断对象是否为垃圾邮件。

对反垃圾邮件功能来说，运行管理也是非常关键的。同样的邮件未必对所有人来说都是垃圾邮件，在某些人看来它也许是有用的，但在另一些人看来它也可能只是垃圾。考虑到这样的情况，我们应根据不同的环境进行适度的调整。

- 内容过滤

内容过滤是一种限定可浏览网站的功能。UTM 将各种网站的网址分门别类（如分为违法性、犯罪性较高的网站，成人色情网站，等等），并将这些网址保存在特定的数据库中。过滤时，将用户浏览的网址和数据库进行对照，然后决定对该网站是允许还是拒绝。

对内容过滤来说，运行管理同样也至关重要。同样的网站未必对所有企业来说都是无用的，反过来，也未必都是有用的。因此，实施这项功能之后我们还应根据不同的环境做一些定制处理。

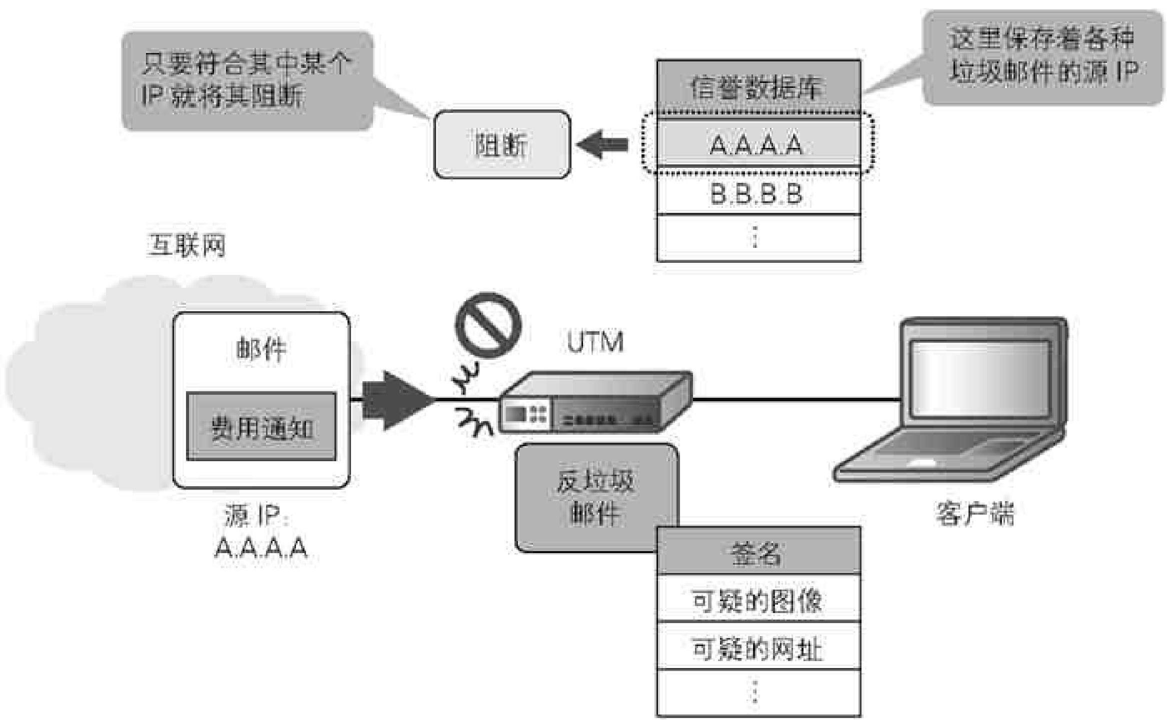


图 **3.1.39**　反垃圾邮件功能基于签名和信誉判断对象是否为垃圾邮件

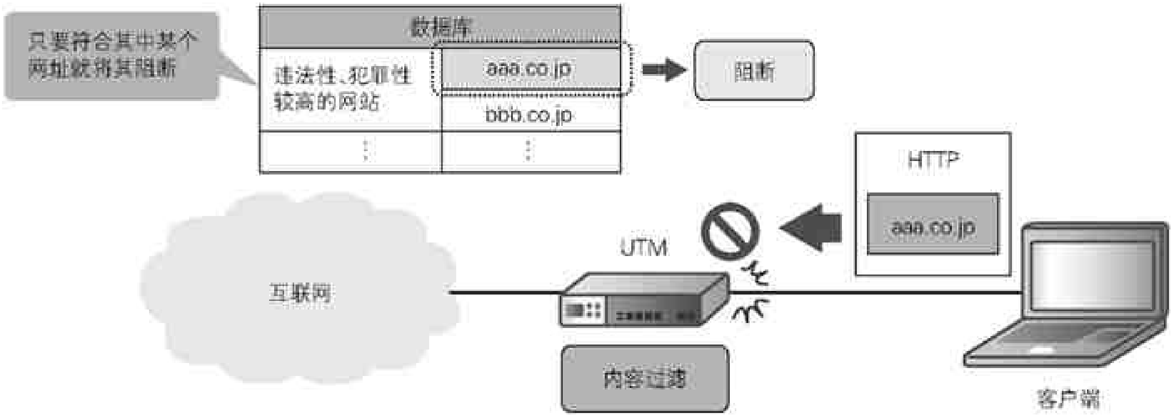


图 **3.1.40**　使用内容过滤功能限定可以浏览的网站

新一代防火墙能灵活控制应用程序并使其可视化

还有一项技术进步，就是新一代防火墙。新一代防火墙大多具备 UTM 功能，所以也可以说它是 UTM 发展进步的一个结果。新一代防火墙有两个特点，一个是能够识别应用程序，另一个是可视化。

- 识别应用程序

  新一代防火墙不是将端口号当作应用程序进行识别，而是根据多个要素去识别应用程序并建立过滤规则。应用程序正在变得越来越复杂。以 HTTP 为例，如今，HTTP 早已不局限于网站浏览，而是在收发文件和实时交换信息等方面也发挥着作用。于是人们不再只单纯地将 TCP/80 当作 HTTP 去分门别类，而是根据网址、内容信息、文件后缀等多种信息对应用程序进行更加深层的细分。举个例子，以往只要允许 HTTP，我们就能够浏览 Facebook、Twitter 这些网站了，然而在新一代防火墙中，即便是同一个 HTTP 通信，我们也可以根据实际需要允许 Facebook 但拒绝 Twitter，像这样基于应用程序对它们分别进行控制。

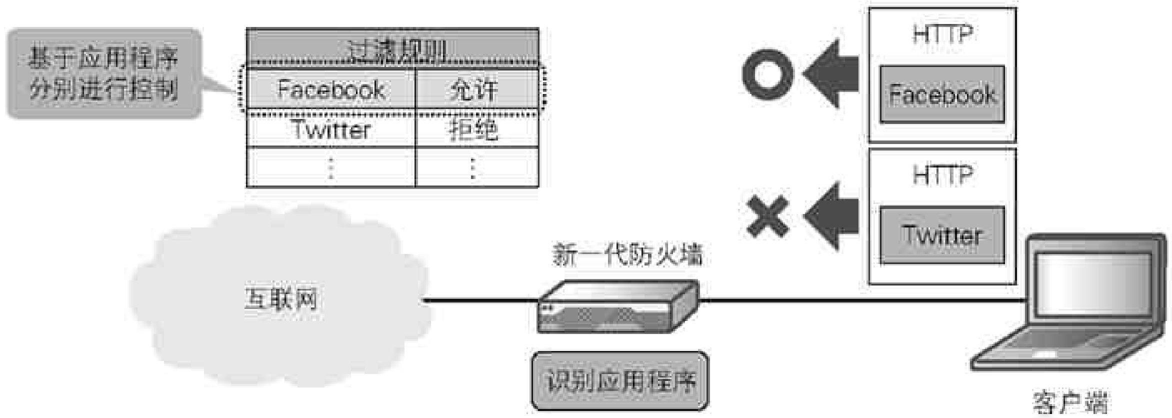


图 **3.1.41**　新一代防火墙能够更加深入地识别应用程序

- 可视化

  新一代防火墙的另一个特点是通信的可视化。对管理人员来说，谁在使用什么应用程序、使用的频率有多高，这些都是非常重要的信息。新一代防火墙能够将识别出来的通信制成图表并显示出来，看起来一目了然。也许有的人会想，可视化就这么点作用啊？可别小看它，实际上这点作用能帮上我们的大忙。作为一项管理作业，获取大量的统计信息并将它们整理成 Excel 表其实比我们想象的要麻烦许多，而且手工去做的话往往只能得到并不完美的图表，利用可视化功能则能够为我们省去这些日常性的管理工作，物有所值。

  防火墙这一行业的发展还处于摸索时期，目前多少有些混沌无序，但笔者相信最合理的状态一定会在某个阶段出现。所以，现阶段我们一定要在弄清需要防火墙的哪些功能、又应如何使用这些功能之后，再去做出选择。

3.1.3　通过负载均衡器分散服务器的负荷

下面介绍在传输层运作的另一种设备——负载均衡器。负载均衡器是一种利用网络层（IP 地址）和传输层（端口号）的信息为多台服务器分配连接的设备。负载均衡器上设有虚拟服务器，虚拟服务器收到连接后按照预先制定的规则将连接分配给各台服务器，以此来分散服务器的处理负荷。

3.1.3.1　目的 **NAT** 是服务器负载均衡技术的基础

目的 NAT 是服务器负载均衡技术的基础。首先，我们就来看看目的 NAT 是如何实现的。

目的 NAT 是根据连接表中的信息执行的，这个连接表和用于防火墙的连接表在要素上有些不太一样。用于负载均衡器的连接表由“源 IP 地址：端口号”“虚拟 IP 地址（转换之前的 IP 地址）：端口号”“实际 IP 地址（转换之后的 IP 地址）：端口号”和“协议”等信息构成，查询该表就能知道什么样的通信会被分散到哪台服务器上去。

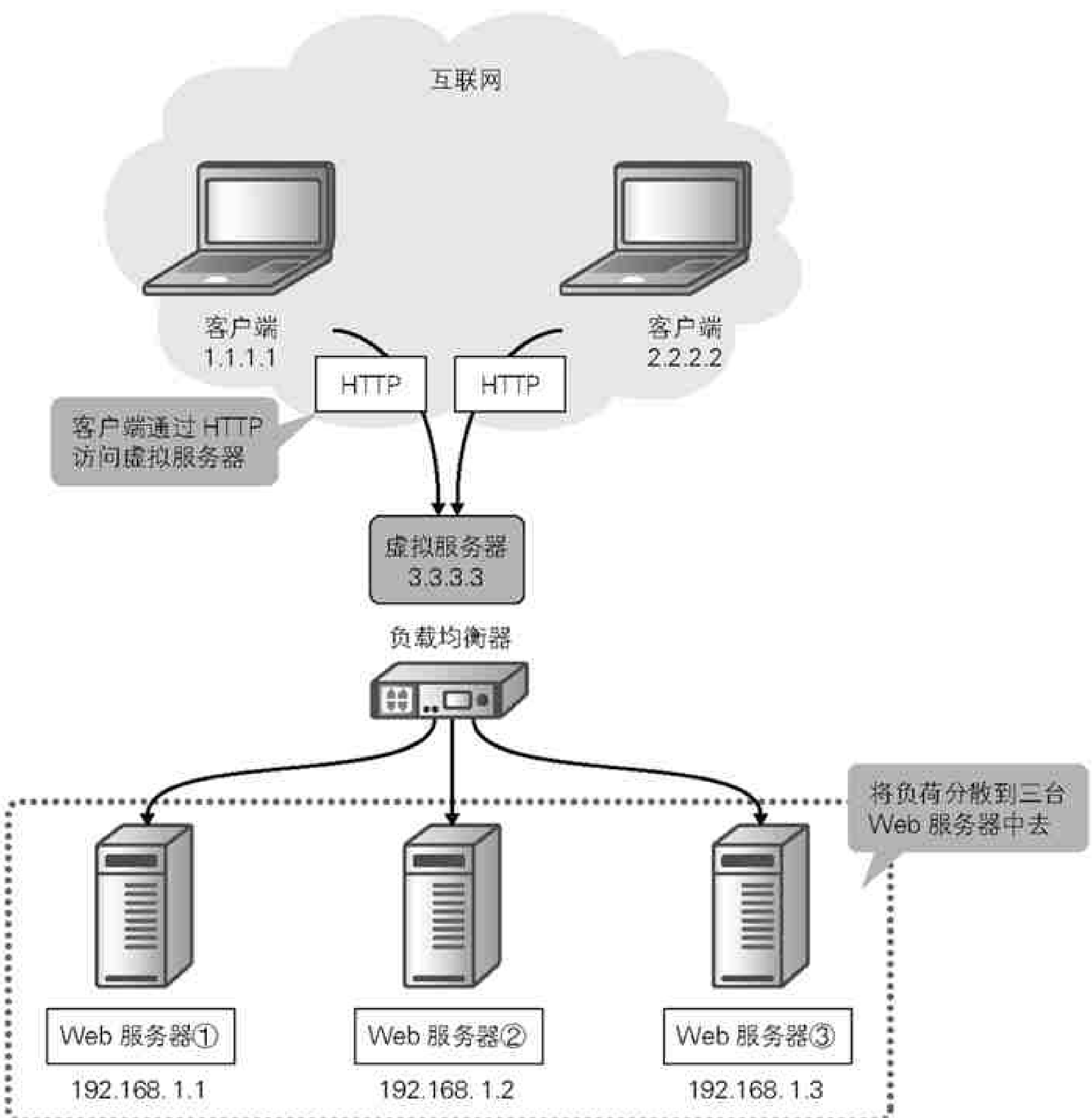


图 **3.1.42**　有助于我们了解服务器负载均衡技术的结构实例

1 →　负载均衡器通过虚拟服务器收到客户端的连接。这时候目的 IP

地址是虚拟服务器的 IP 地址，也就是虚拟的 IP 地址。收到的这个链接在连接表中进行管理。

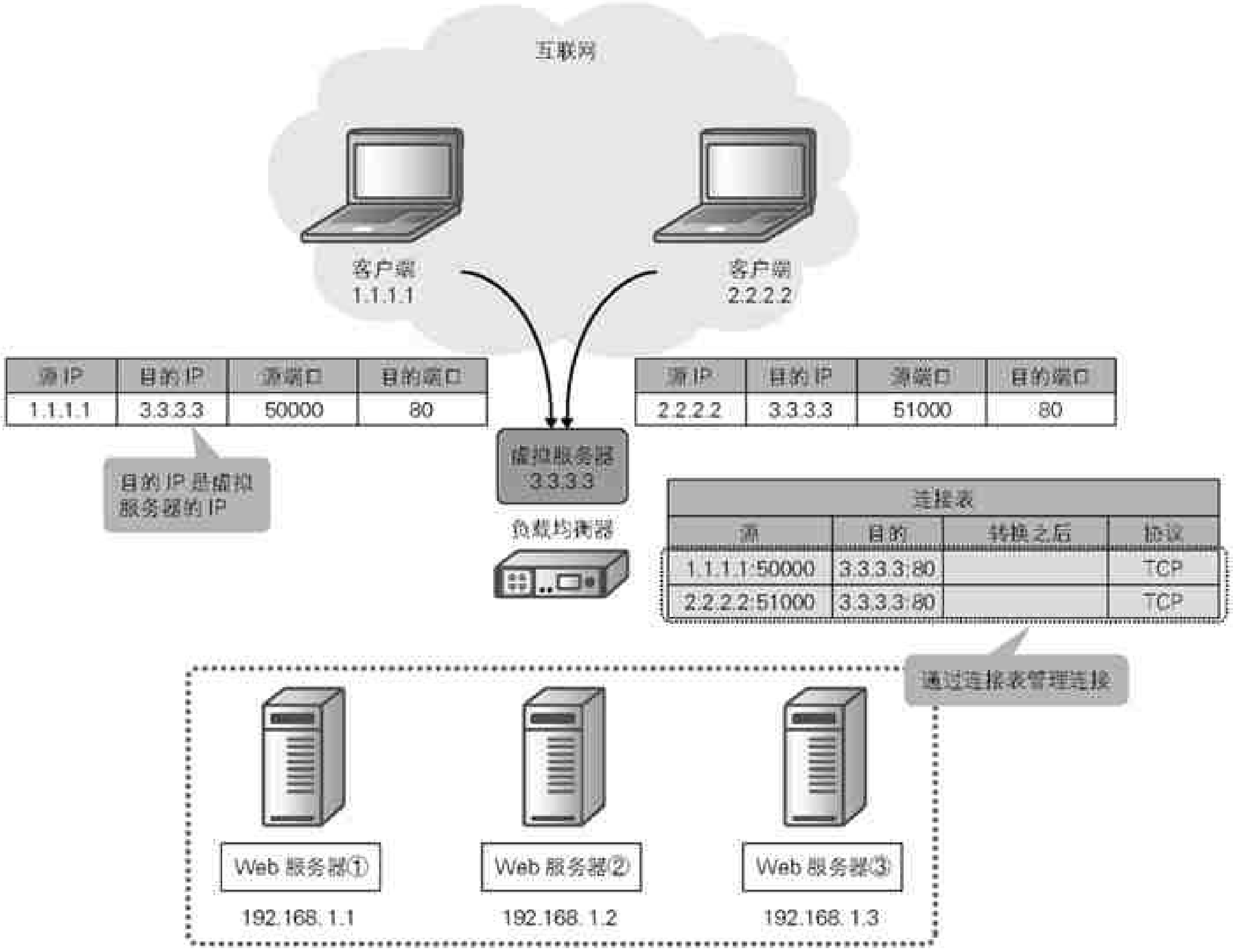


图 **3.1.43**　客户端访问虚拟服务器

2 →　负载均衡器将目的 IP 地址（虚拟 IP 地址）转换为与之对应的负载均衡服务器的实际 IP 地址。转换的实际 IP 地址能够基于服务器状态、连接状态等各种因素动态地改变，因此连接得以分散到各处。转换之后的实际 IP 地址也会被写入连接表，在连接表中接受管理。

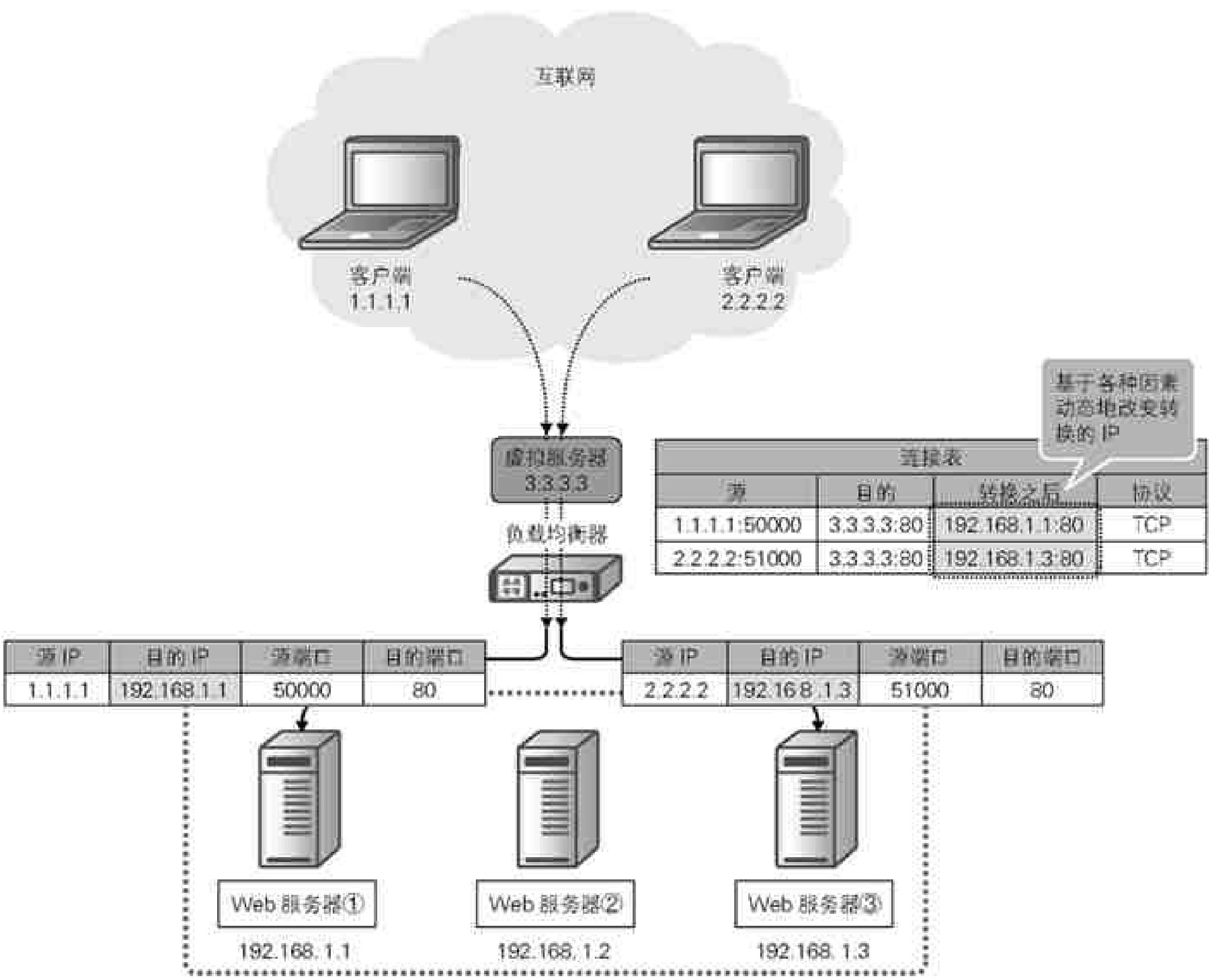


图 **3.1.44**　负载均衡器将虚拟的目的 **IP** 转换为实际的 **IP** 地址

3 →　接下来我们再看返回的通信。服务器收到连接后对其进行应用处理，然后会将处理结果返回给已成为默认网关的负载均衡器。这里负载均衡器要做一个和发送时恰好相反的转换处理，也就是要对源 IP 地址（前面是针对目的地址）执行 NAT 处理。负载均衡器在连接表中管理发送通信的信息，并根据该信息向客户端返回通信。

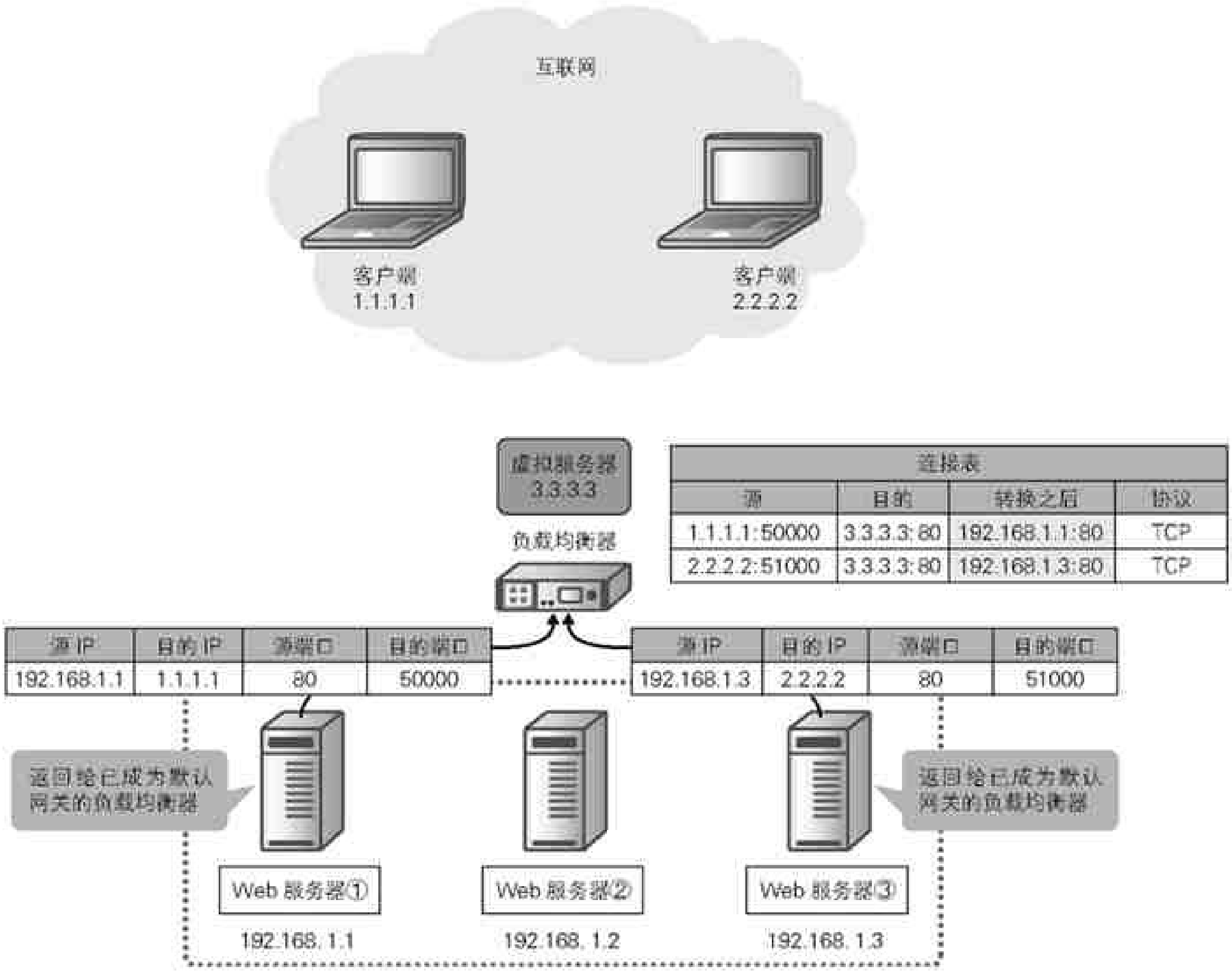


图 **3.1.45**　返回给负载均衡器

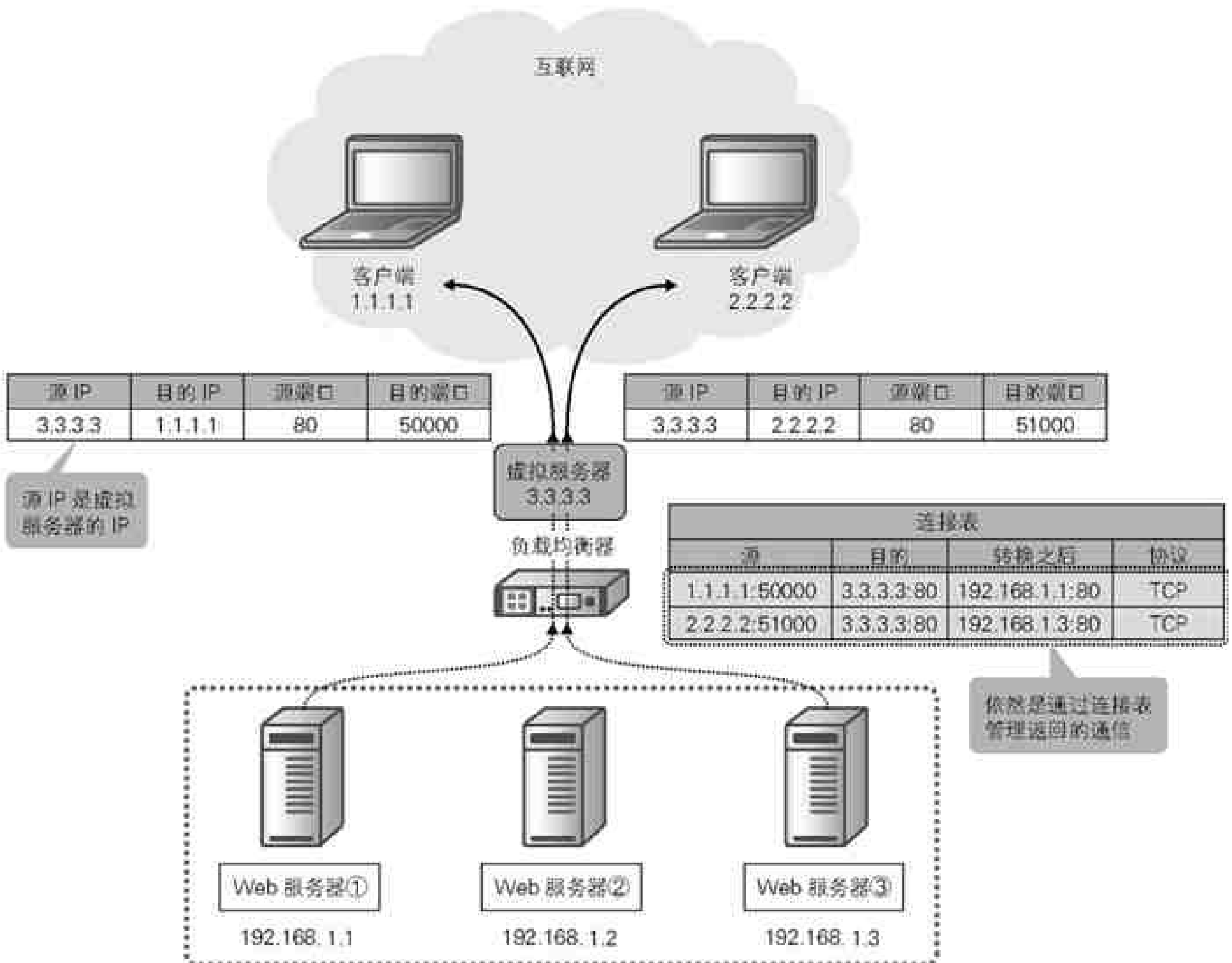


图 **3.1.46**　参照连接表转换源 **IP**

3.1.3.2　通过健康检查监控服务器的状态

转换的实际 IP 地址取决于两个因素，一个是健康检查，另一个是负载均衡方式。我们就是要利用这两项功能，来决定将虚拟 IP 地址转换成哪个真实的 IP 地址。

- 通过三种健康检查监控服务器

  健康检查这种功能用来监控负载均衡服务器的状态。我们知道，将连接分配给已经宕机的服务器是没有意义的，它不会给出任何响应。为了避免出现这种毫无意义的局面，负载均衡器会通过监控报文定期检查服务器是否正常运作，一旦测出宕机，就会将该服务器排除到负载均衡对象之外。有些生产商称之为“健康监控”或“（服

务器）探测”什么的，都是指同一种功能。健康检查大致可分为 L3 检查、L4 检查和 L7 检查三种，分别是针对不同的层实施检查。

- **L3** 检查

L3 检查通过 ICMP 来检查 IP 地址是否正常。假设未做冗余配置的 NIC 出现了故障或者缆线发生断离，服务器就无法返回 ICMP 包，这时候负载均衡器会将该服务器隔离到负载均衡对象之外。如果 L3 检查未能过关，L4 以上的所有服务就无法提供，因此，L4 检查和 L7 检查也将无法过关。

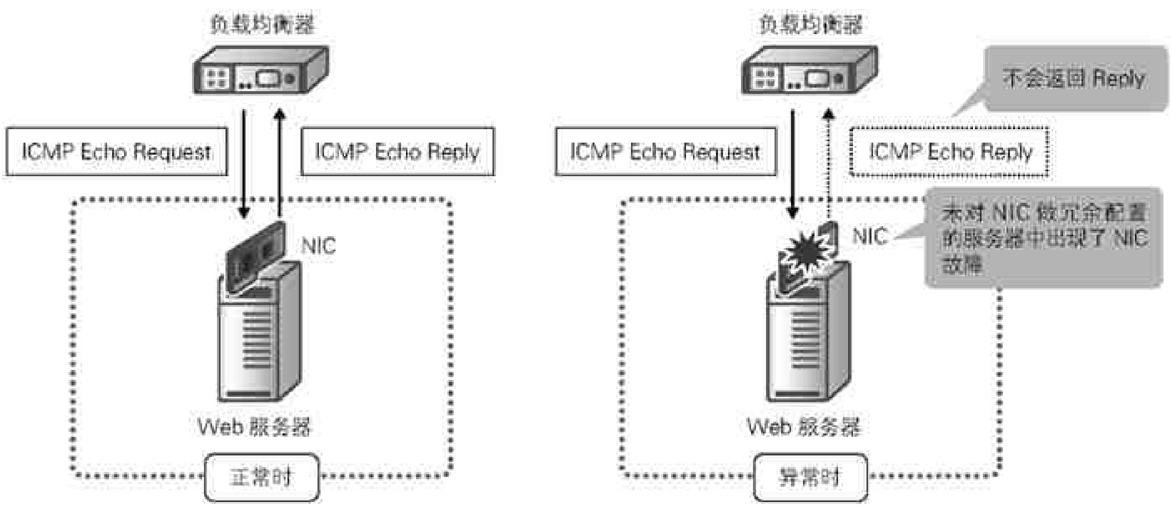


图 **3.1.47**　**L3** 检查是针对 **IP** 地址的检查

- **L4** 检查

L4 检查通过三次握手来检查端口号是否正常。假设在 Web 服务器中默认使用的是 TCP/80 端口，负载均衡器会对该端口定期执行三次握手以确认响应的情况。如果 IIS 或 Apache 的进程中断，TCP/80 的响应就会中断，这时负载均衡器会将该服务器隔离到负载均衡对象之外。应用程序是在服务器进程上工作的，因此，如果 L4 检查未能过关，那么 L7 检查也将无法过关。

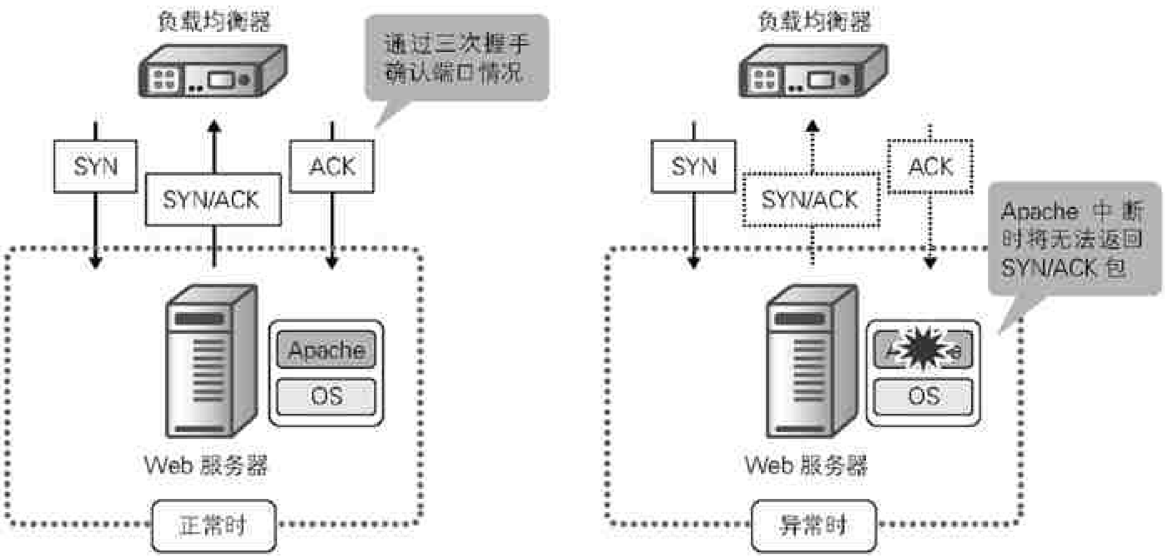


图 **3.1.48**　**L4** 检查是针对端口号的检查

- **L7** 检查

L7 检查中的 L7 指的是第七层，也就是应用层。L7 检查通过真实的应用通信来检查应用程序是否正常。例如，网络应用程序的状态是以状态码的形式返回的，负载均衡器监控的就是这个状态码。网络应用程序如果出现故障，就会返回一个表示异常情况的状态码，这时负载均衡器就会将该服务器隔离到负载均衡对象之外。

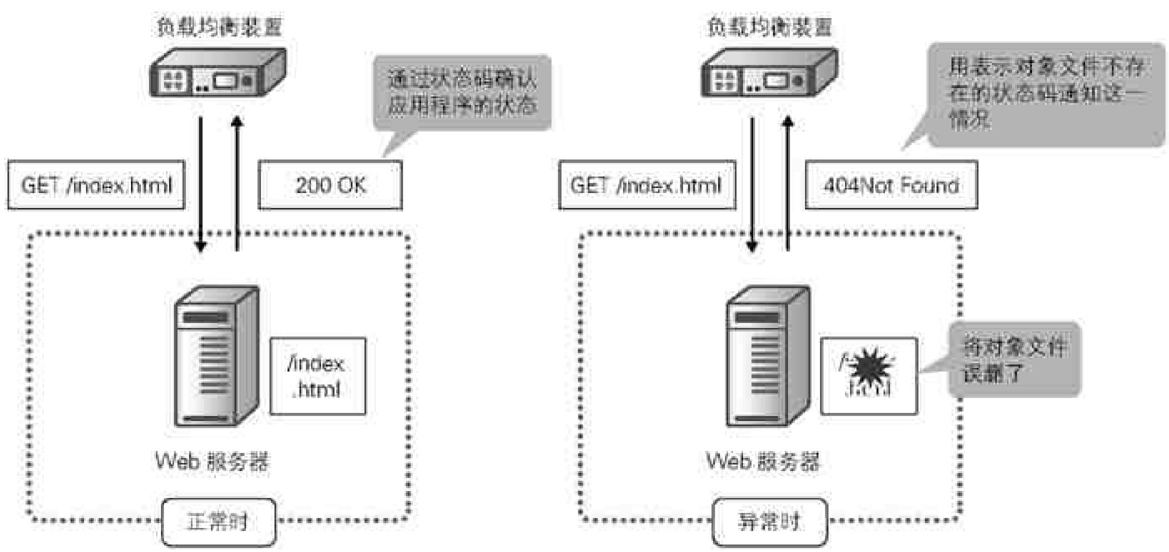


图 **3.1.49**　**L7** 检查是针对应用程序的检查

要根据应用程序和服务器规格变换负载均衡方式

负载均衡方式指的是“参考什么信息，将负荷分散到哪些服务器上去”。负载均衡方式不同，通过目的 NAT 转换的目的 IP 地址也就不同。

负载均衡方式大致可以分为静态和动态两种。静态负载均衡方式根据预先的设置来决定将负荷分散到哪些服务器上去，并不关心服务器的实际状况如何；动态负载均衡方式则根据服务器的实际状况决定将负荷分散到哪些服务器上去。实际上，负载均衡方式有更多更细的分类，适用于各种不同的设备，不过本书仅介绍它们当中具有代表性的两个细化分类。

表 **3.1.6**　负载均衡方式分为静态和动态两种

分类	负载均衡方式	解说
静态	轮询	按照顺序分散负荷
	加权和比例	根据加权和比例分散负荷
动态	最少连接数	将负荷分散给连接数最少的服务器
	最短响应时间	将负荷分散给响应速度最快的服务器

- 轮询

  轮询方式将收到的请求（Request）按照顺序分配给负载均衡服务器，属于静态负载均衡方式。具体说来就是依次分配给服务器①、服务器②、服务器③……原理非常简单，所以人们容易预测到下一个连接，管理起来也很方便。当负载均衡服务器的规格都一样并且每次处理的时间都较短时，轮询方式是极有用武之地的。然而如果不是这样的环境，比如服务器的规格参差不齐，或者该环境需要能够保持会话（会话保持功能），那我们就不能不假思索地分配连接了，否则负载均衡的效率会非常糟糕，这一点一定要注意。

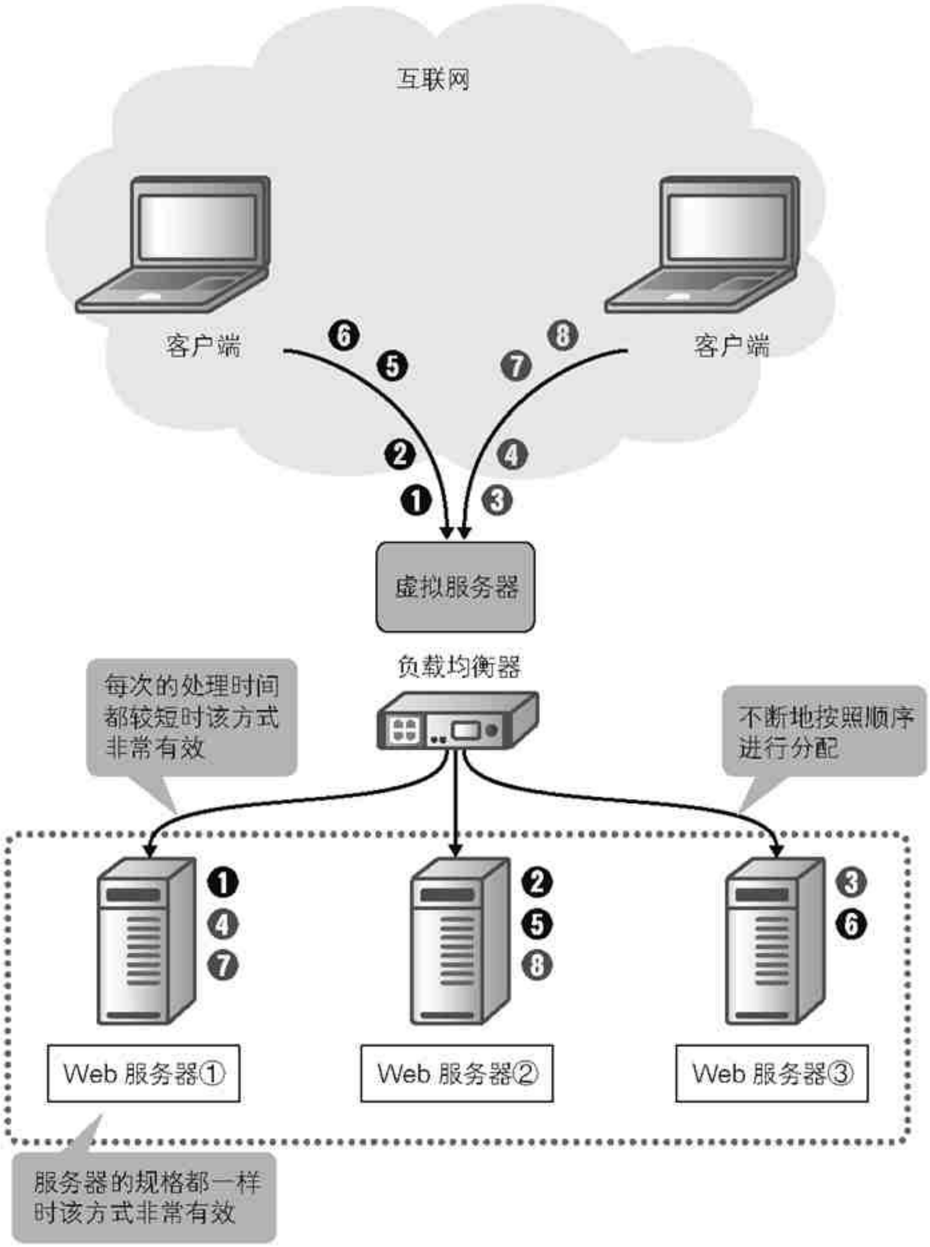


图 **3.1.50**　轮询方式是按照顺序分配

- 加权和比例

加权和比例方式预先给每台服务器设置好比例，然后根据整体的比例情况分配连接，属于静态负载均衡方式。采用轮询方式时，服务器即使规格较低也能够分到连接。而加权和比例方式则是将规格较高的服务器设置为高比例，将规格较低的服务器设置为低比例，其结果是比例较高的服务器能够优先获得连接分配。在负载均衡服务

器规格参差不齐的环境里，这种方式能够发挥较大的作用。

加权和比例方式常常作为各种负载均衡方式的一个可选项出现，当环境中的服务器规格参差不齐时，人们往往将它和其他负载均衡方式一起使用以达到最佳效果。

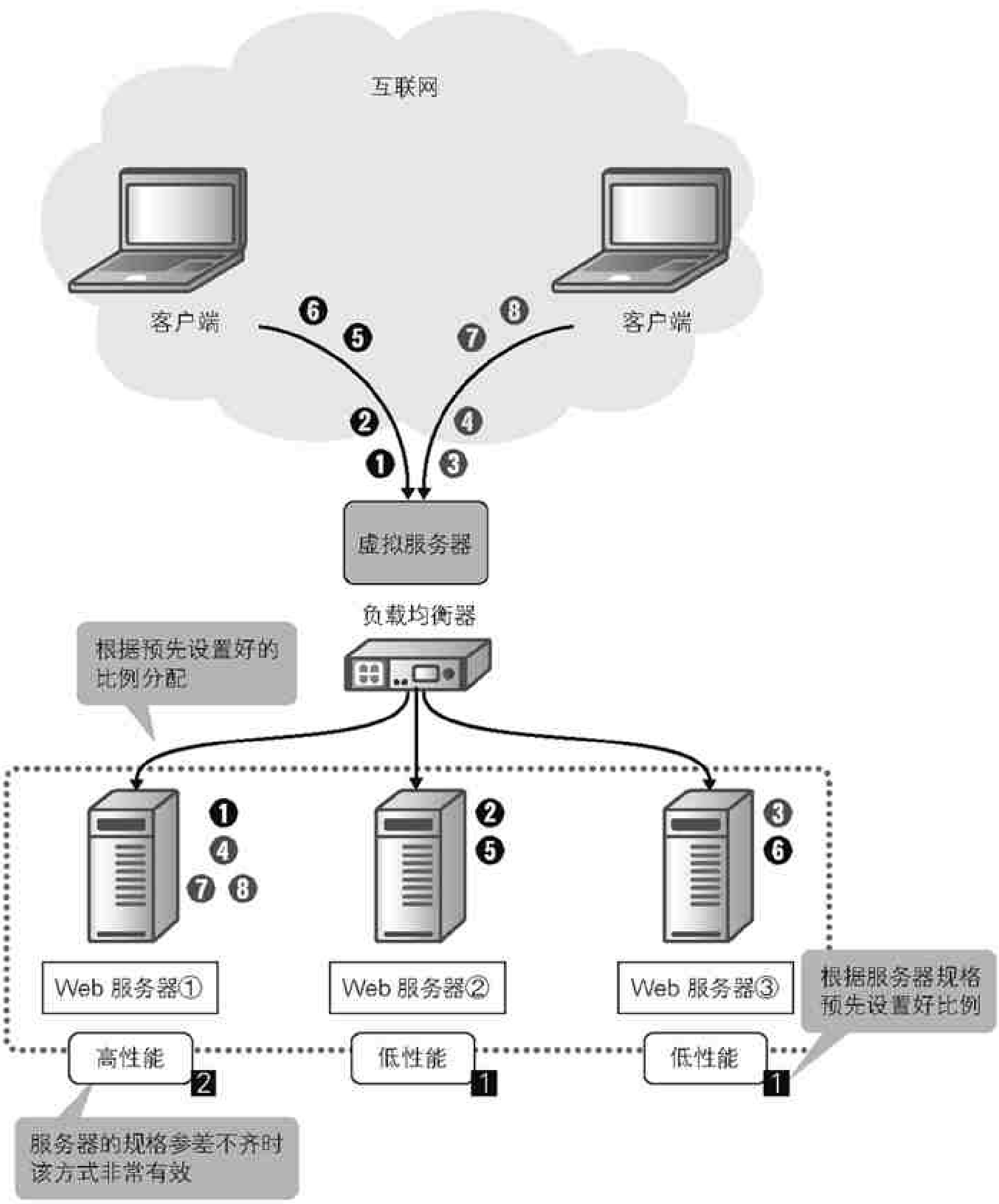


图 **3.1.51**　根据比例分配

- 最少连接数

最少连接数方式将连接分配给当前联机数量最少的服务器，属于动态负载均衡方式。采用这种方式时，负载均衡器会监测负载均衡服务器的联机数量，选择收到连接时联机数量最少的，也就是处理负荷最小的那台服务器，然后将连接分配给它。

在下面这两种环境中进行负载均衡时，采用这种方式非常有效：应用程序为需要长时间联机的 HTTP/1.1 或 FTP 等；要保持会话（使用会话保持功能）以保证在一定时间内一直向同一台服务器传送信息。

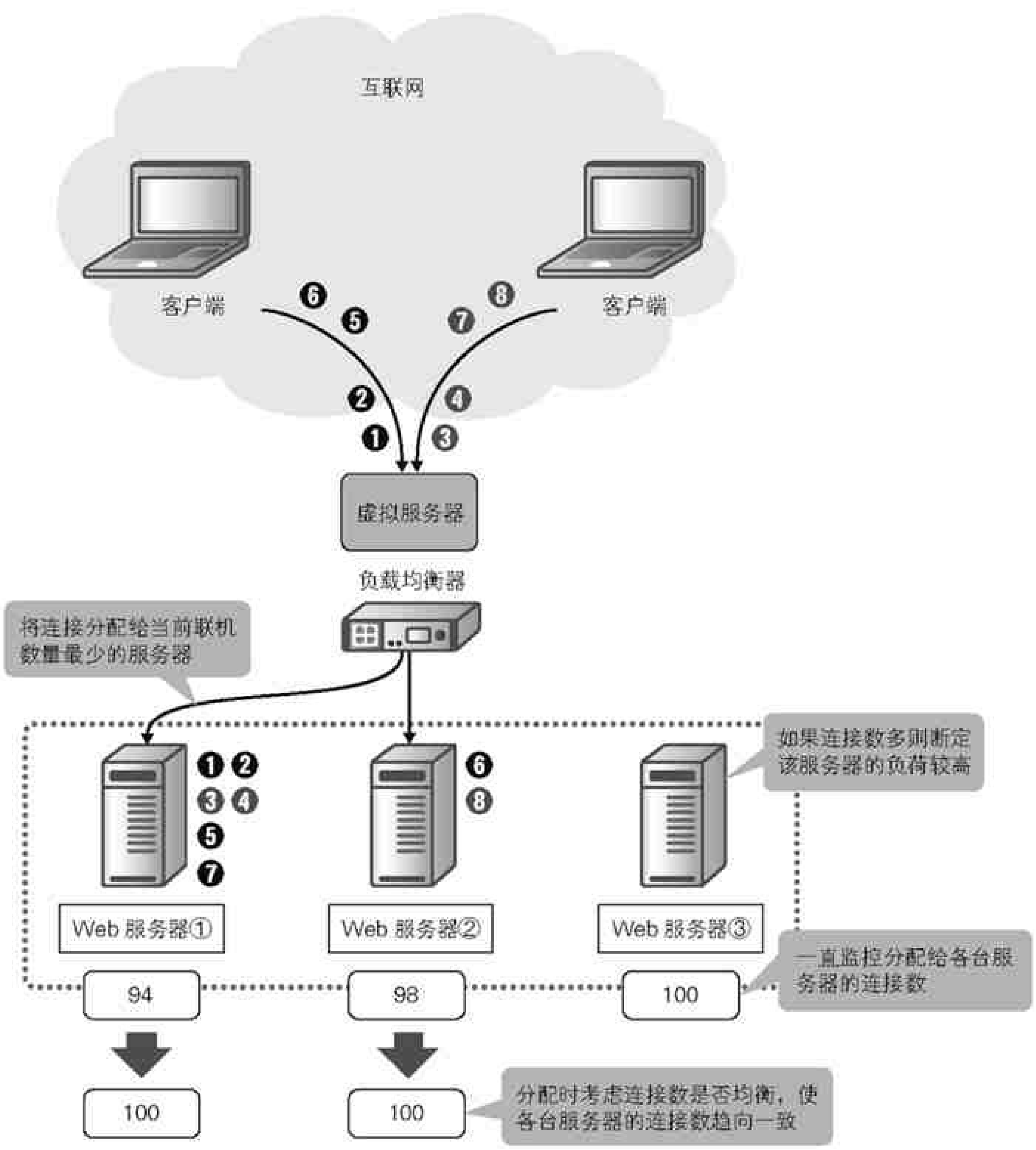


图 **3.1.52**　分配时考虑连接数是否均衡，使各台服务器的连接数

趋向一致

- 最短响应时间

  最短响应时间方式将连接分配给响应最快的服务器，属于动态负载均衡方式。无论是什么服务器，一旦无法处理全量信息，反应就会变得迟钝，最短响应时间方式利用的就是这个原理。采用这种方式时，负载均衡器会根据客户端的请求和服务器的回复去检查服务器的响应时间，选择收到连接时响应最快的那台服务器，然后将连接分配给它。由于这种方式能够根据服务器的处理负荷进行合理的负载均衡，所以在服务器规格参差不齐的环境中非常有效。

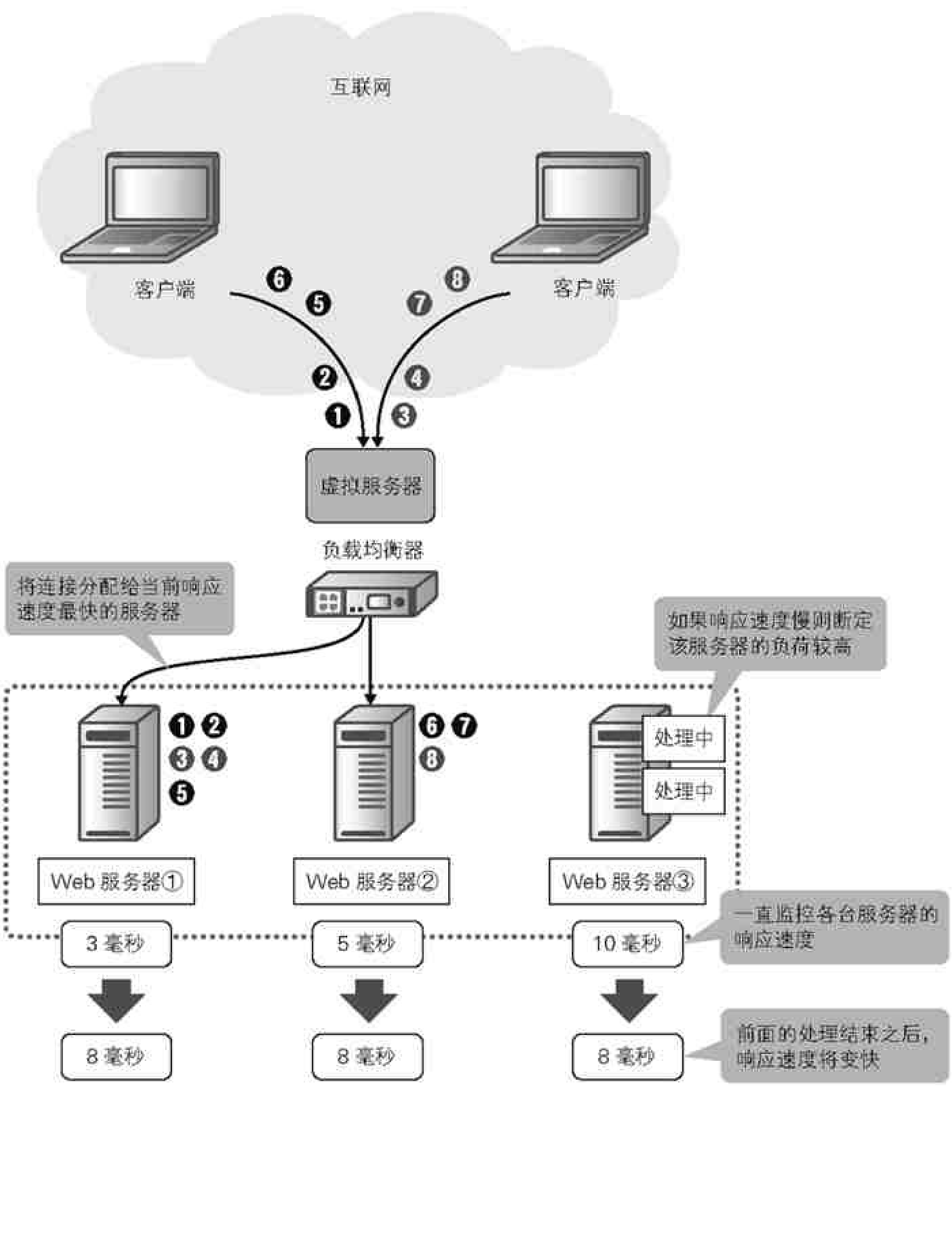


图 **3.1.53**　将连接分配给响应速度最快的服务器

通过会话保持将会话持续分配给同一台服务器

会话保持功能将应用程序的同一会话持续分配给同一台服务器。也许你会想，既然是一种负载均衡技术，那为什么会持续分配给同一台服务器呢？这不矛盾吗？事实上，从全局出发来看，这种方式的确能够起到均衡负载的作用。

对某些应用程序来说，如果不将一系列的处理都放在同一台服务器中进行，就无法保证该处理能够前后呼应，购物网站就是一个很好的例子。在购物网站，“放入购物车”“付款”这一系列的处理必须在同一台服务器上进行，不可能在服务器①中将商品放入购物车却在服务器②中付款。如果“放入购物车”是在服务器①中处理的，那么付款也必须仍在服务器①中处理。这时候就要求我们使用会话保持功能，它能根据特定的信息将会话持续分配给同一台服务器，使“放入购物车”和“付款”这一系列的处理都能在同一台服务器中完成。

支持会话保持功能的后台是会话保持表[3]。会话保持表记录着连接中的特定信息以及分配的服务器，使得后续的连接能够依然分配给同一台服务器。会话保持表中记录的信息多种多样，具体记录什么信息则取决于记录方式。下面，本书就介绍两种比较常用的会话保持功能。

> [3] 有些设备并不是通过会话保持表去处理 Cookie 会话保持（Insert 模式）的，请仔细确认设备的设计规格。

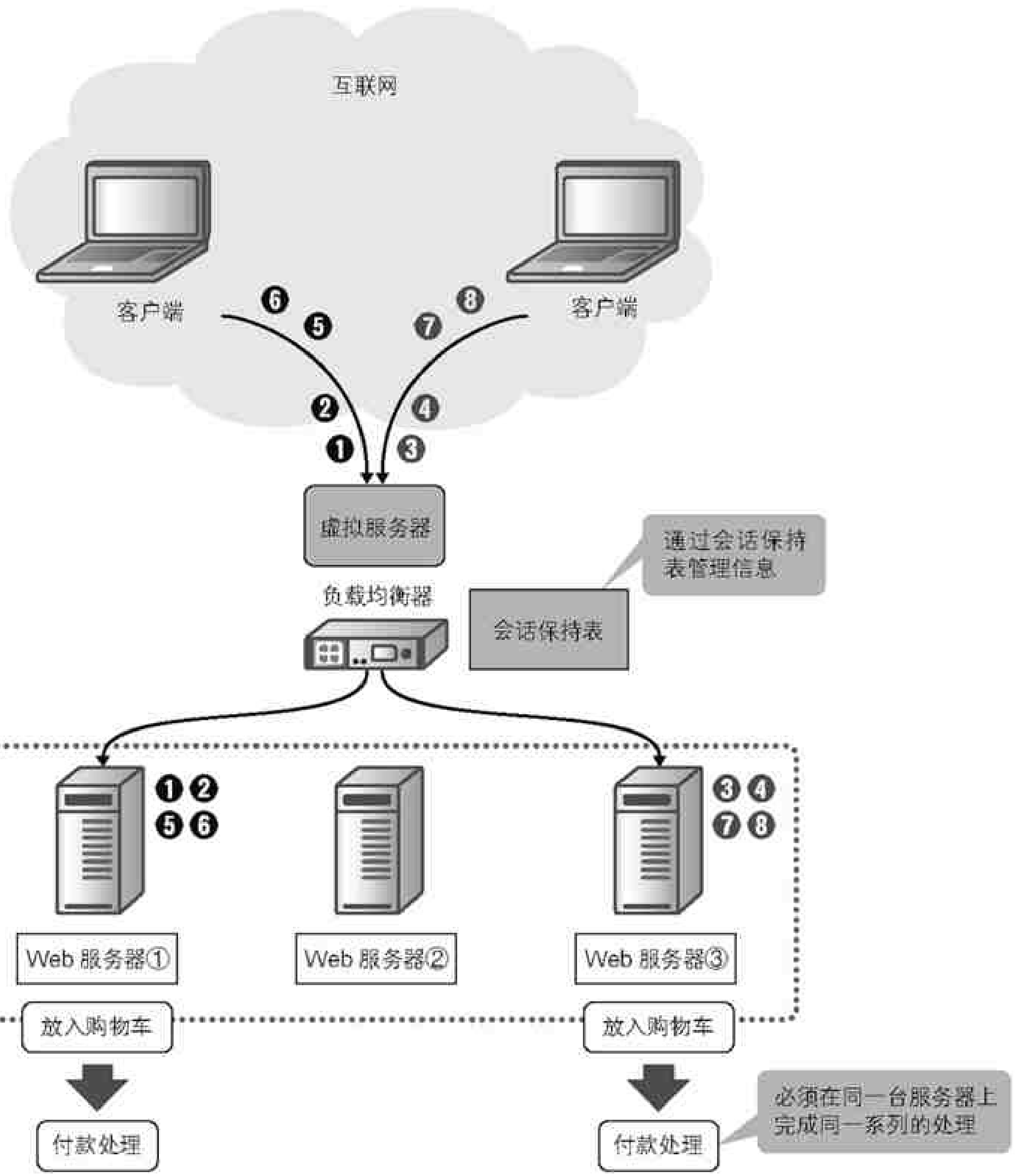


图 **3.1.54**　通过会话保持将会话持续分配给同一台服务器

- 源 **IP** 地址会话保持

  正如其名，源 IP 地址会话保持是一种基于源 IP 地址将会话持续分配给同一台服务器的会话保持。这个应该很容易理解，例如，当源 IP 地址是 1.1.1.1 时将会话持续分配给服务器①，当源 IP 地址是 2.2.2.2 时将会话持续分配给服务器②。如果对方是在互联网上公开的虚拟服务器，那么，由于连接的源 IP 地址是全世界的全球 IP 地址，所以从全局上看它的确能够起到分散负荷的作用。我们设置分配持续时间时应考虑应用程序的时效，一般来说，设置得比应用程

序的时效稍微长一点就能保证处理能够前后呼应、协调一致。

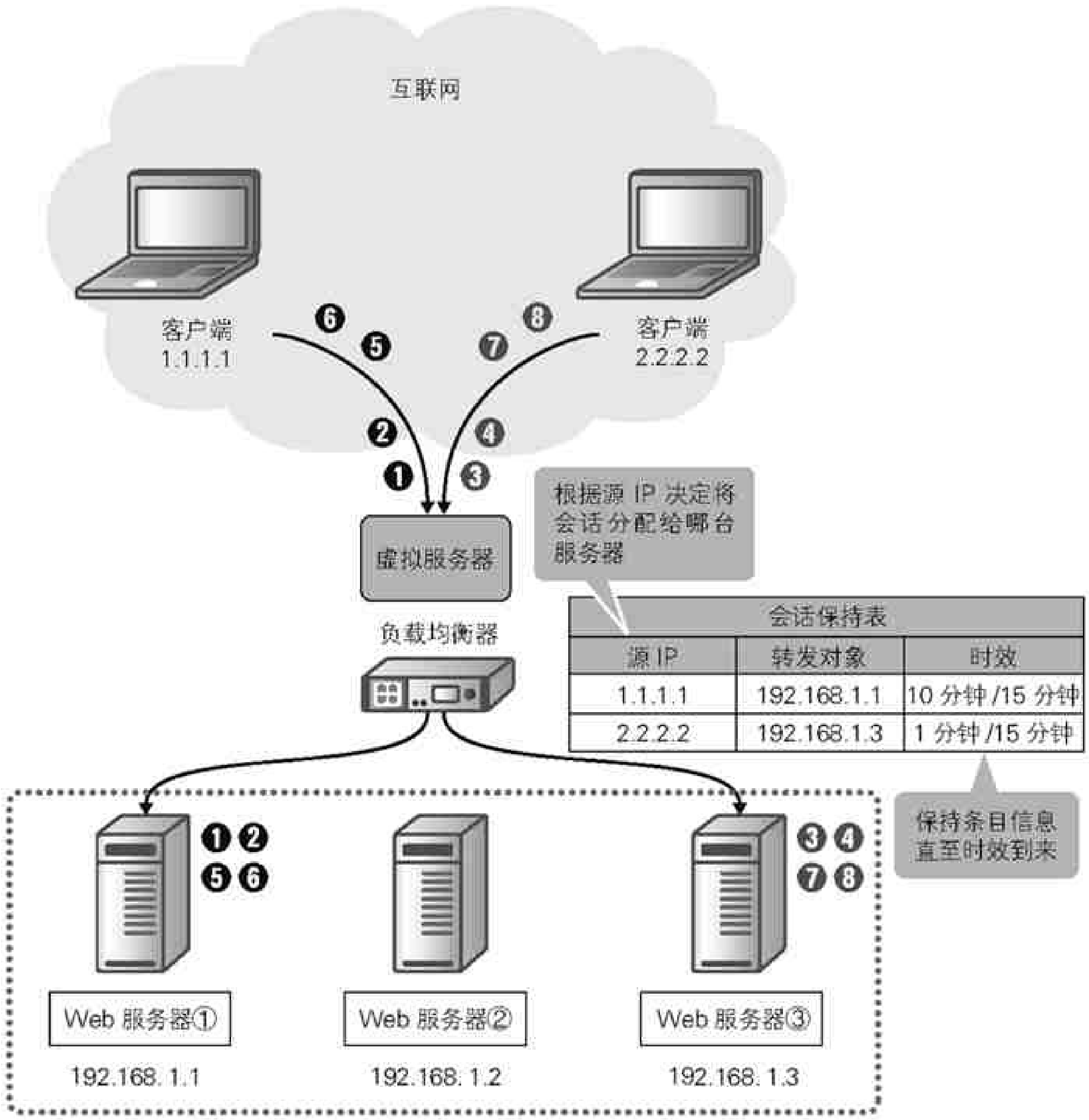


图 **3.1.55**　如果源 **IP** 地址相同，就将会话分配给相同的服务器

源 IP 地址会话保持非常容易理解，管理起来也十分方便。然而，它有一个足以致命的弱点，那就是在多个客户端共用一个源 IP 地址的环境（如 NAPT 环境或代理环境等）中无法分散负荷。假设某个 NAPT 环境拥有 1000 台客户端，那么这 1000 台客户端的负荷都会被分配给同一台服务器，这样的话负载均衡根本无从谈起。源 IP 地址会话保持对环境是非常挑剔的，因此我们一定要仔细确认连接

环境之后再去决定是否使用。

- **Cookie** 会话保持（**Insert** 模式）

  Cookie 会话保持（Insert 模式）是基于 Cookie 的信息将负荷持续分配给同一台服务器的会话保持，仅在使用 HTTP 或 SSL 加速的 HTTPS 环境中有效。

  Cookie 指的是通过与 HTTP 服务器之间的通信，将特定信息暂时保存到浏览器中的一种机制，同时也指保存这些信息的文件。每个 FQDN（Fully Qualified Domain Name，完全合格域名）都拥有相应的 Cookie。最开始的 HTTP 回复决定了将负荷分配给哪台服务器，这些信息就包含在 Cookie 之中，由负载均衡器交给客户端[4]。由于后续的 HTTP 请求中都带有 Cookie，所以负载均衡器查看这些 Cookie 信息就能将负荷持续分配给同一台服务器。

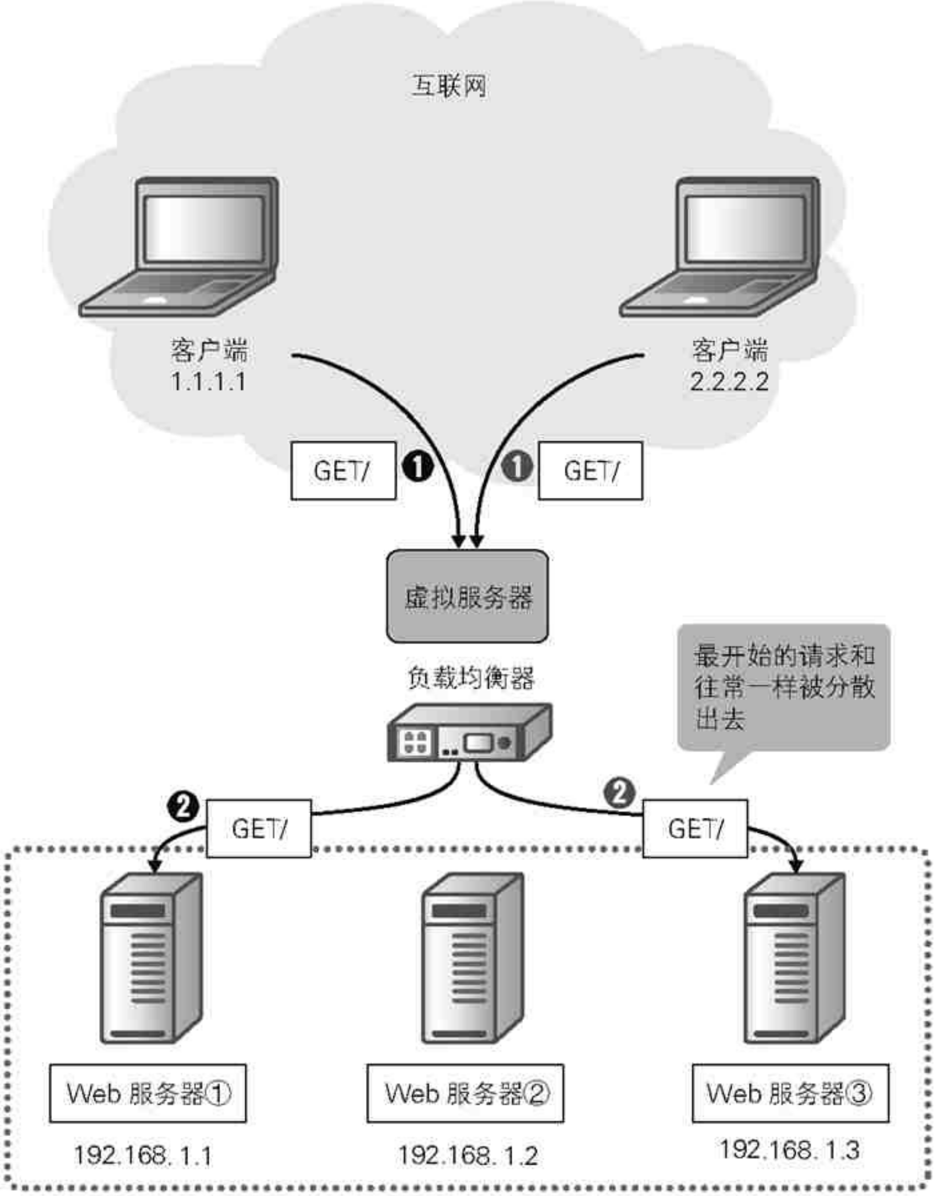


图 **3.1.56**　最开始的请求和往常一样被分散出去

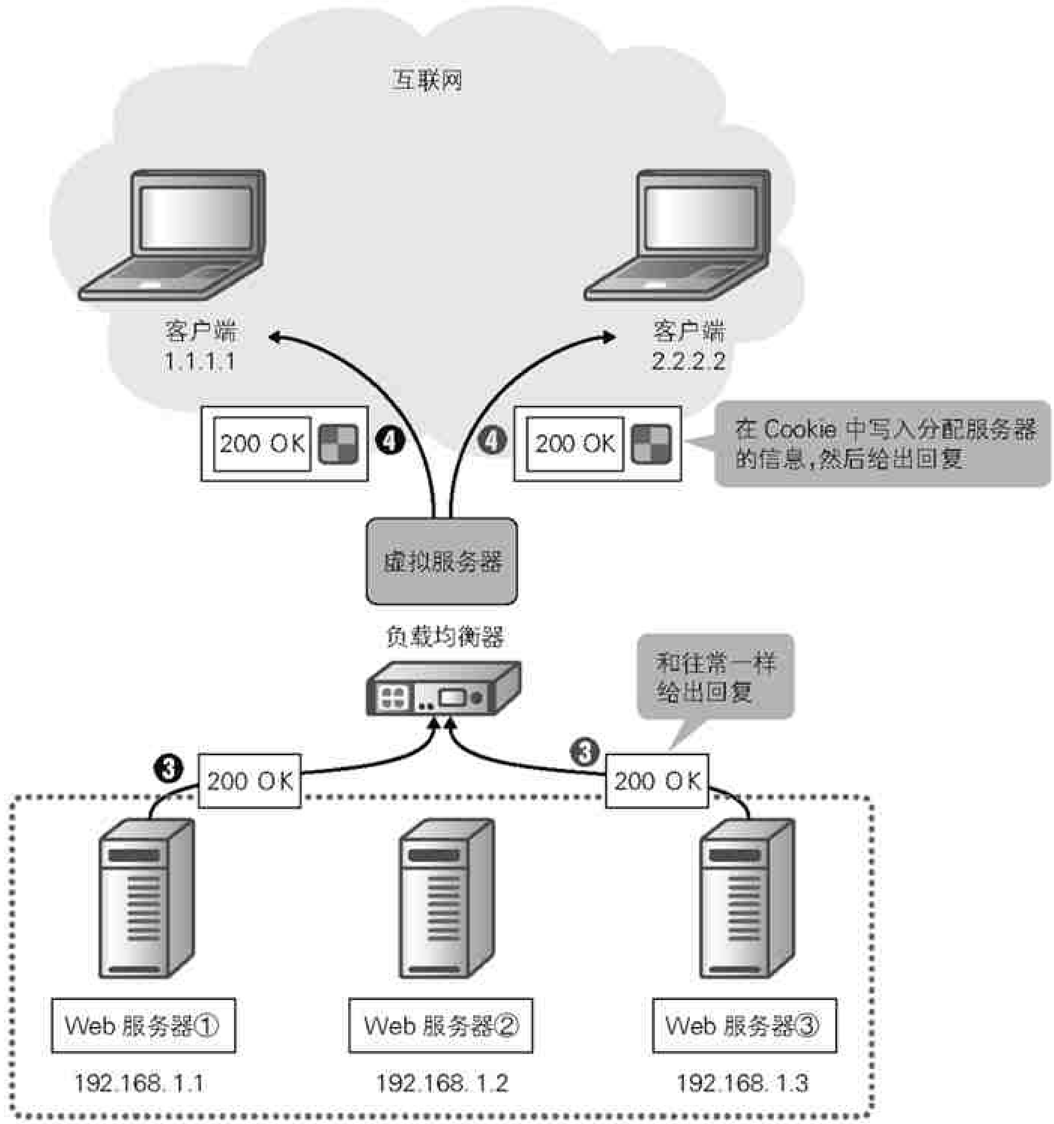


图 **3.1.57**　在最开始的回复中装入 **Cookie**

Cookie 会话保持远比源 IP 地址会话保持灵活，然而它也有不足之处。首先浏览器必须能够接受 Cookie，其次还需要在应用程序的层面执行一个添加 Cookie 的处理，而这些也会生成一定的处理负荷。

所以，无论采用哪种会话保持都是优缺点并存的。我们在选择的时候应从应用程序的设计、浏览器环境和客户端连接环境等多个角度去考察具体情况，然后再做出判断。

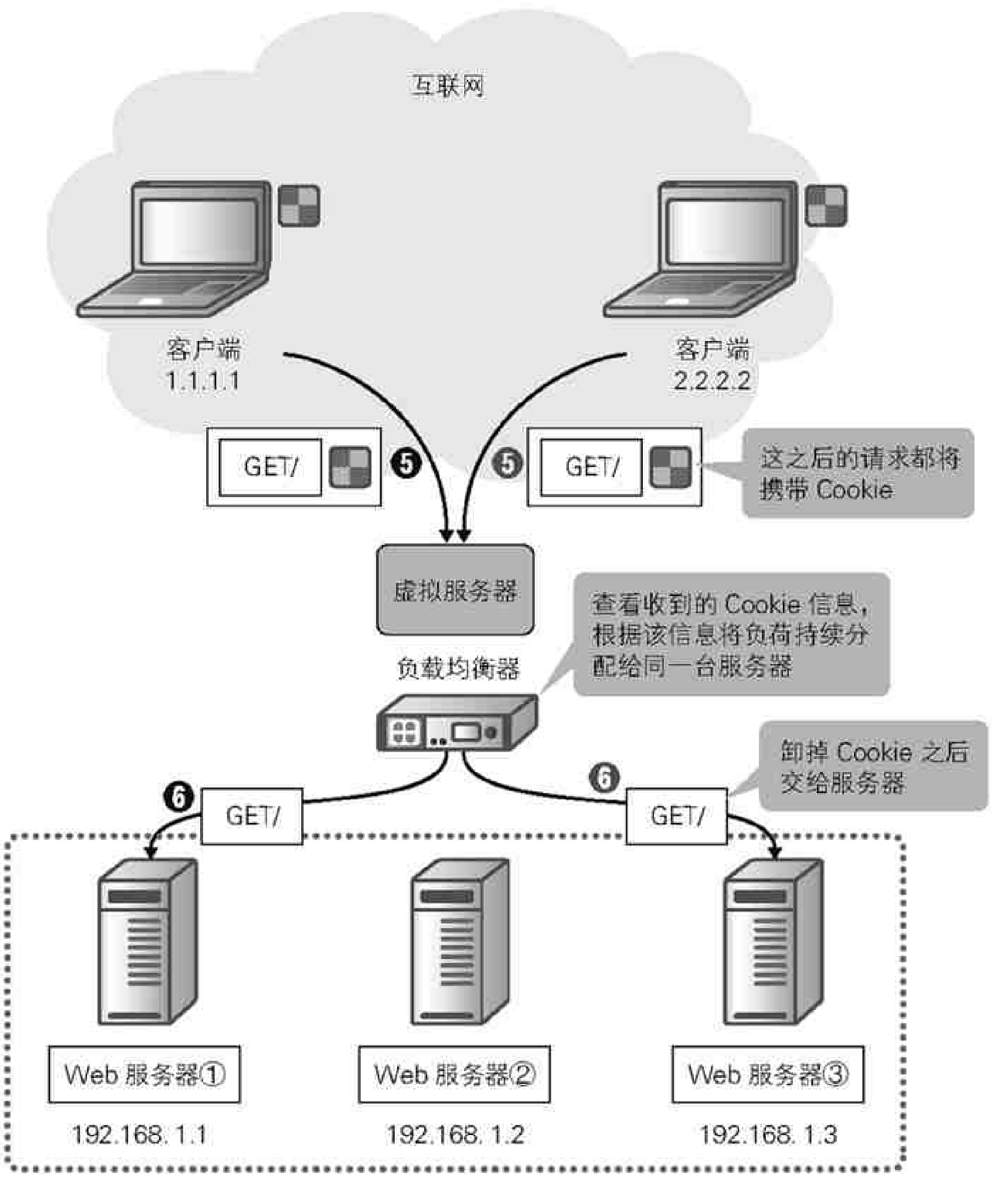


图 **3.1.58**　后续的请求中都带有 **Cookie**

> [4] 实际上是将这些信息作为 HTTP 报头插入。

3.1.3.3　熟练掌握可选功能

最近，负载均衡器的使用范围逐渐延伸到了应用层，其本身也作为“应用交付控制器”获得了人们的广泛认可。在它背后起着支撑作用的并不是负载均衡技术本身，而是丰富多彩的可选功能，本书将从中选出三项来进行说明，分别是 SSL 加速功能、HTTP 压缩功能和连接汇集功能。

SSL 加速功能可代为执行 **SSL** 处理

SSL 加速是一种帮助服务器分散 SSL（Secure Socket Layer，安全套接层）处理负荷的功能。SSL 是一种实现加密和解密的技术，能防止通信被篡改或者遭遇窃听。

为了给通信加密或解密，SSL 中进行着大量的处理 **5** ，这些也都会成为服务器的负荷，于是人们想到了让负载均衡器去直接执行这些处理的办法。客户端只需像往常一样通过 HTTPS（HTTP over SSL，安全超文本传输协议）发出请求即可 **6** 。负载均衡器收到请求之后在前端执行 SSL 处理，将信息作为 HTTP 交给位于后端的负载均衡服务器。这样服务器就不再需要执行 SSL 处理了，负荷得以大大减轻，于是在全局上实现了负载均衡。

5 关于 SSL 的内容将在 3.2.2 节中详细解说。

6 HTTPS 指的是经过 SSL 加密的 HTTP。

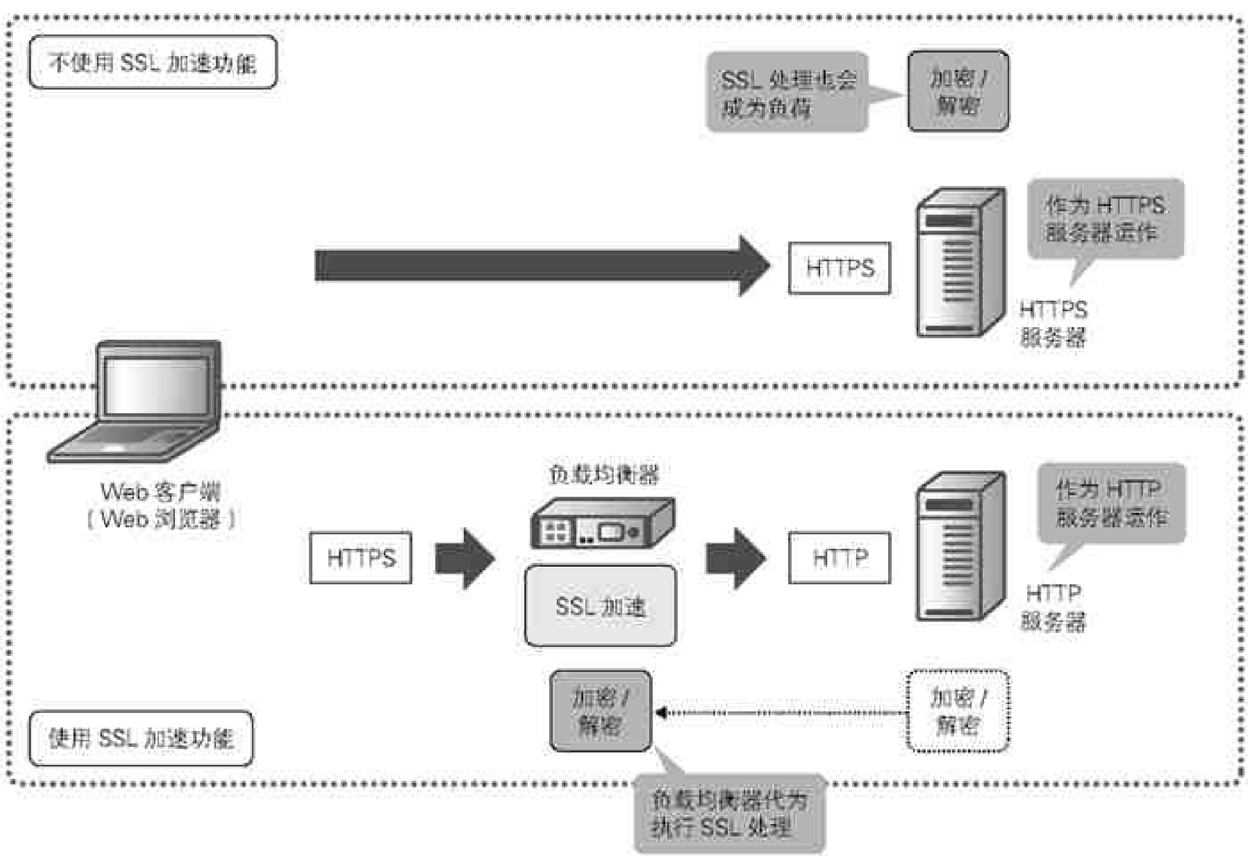


图 **3.1.59　SSL** 加速功能可代为执行 **SSL** 处理

通过 **HTTP** 压缩功能有效地利用带宽

HTTP 压缩功能是指由负载均衡器代替服务器进行 HTTP 压缩处理的一项功能。HTTP 压缩是一种利用 HTTP 对交换的内容进行压缩的技术，可以实现高速传送和低带宽。

HTTP 是一种非对称通信，来自服务器的数据量要比来自客户端的数据量多。负载均衡器将来自服务器的 HTTP 数据压缩之后交给客户端，压缩时只选择容易被压缩的内容（.html、.js、css 等），以使压缩效果达到最大化。

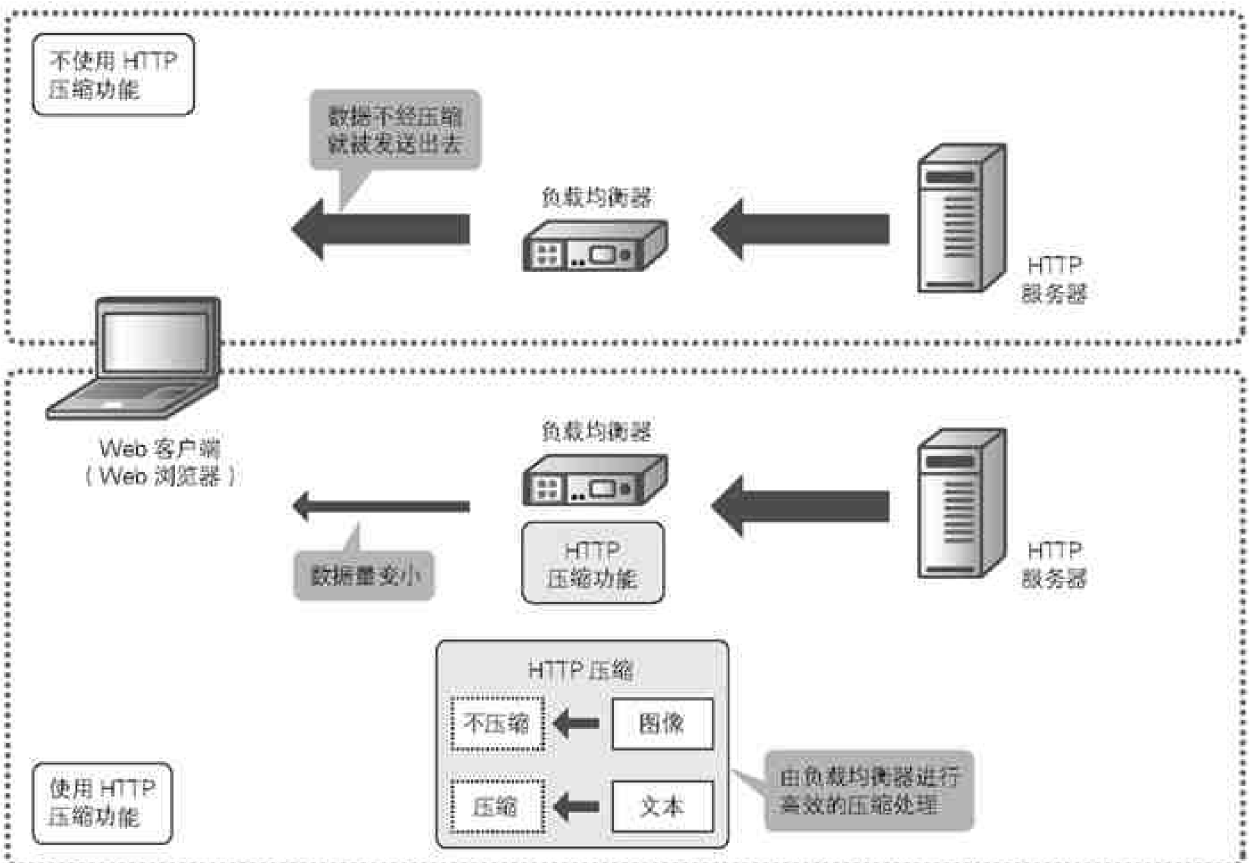


图 **3.1.60　HTTP** 压缩功能可以实现高速传送和低带宽

另外，我们也可以在服务器上激活 HTTP 压缩功能，但是那样就会成为一种基于软件的处理，可能会给 CPU 带来较大的负荷。负载均衡器的 HTTP 压缩功能能够减轻该处理负荷，具有全局意义上的负载均衡效

果。

通过连接汇集功能减轻服务器的负荷

连接汇集是通过负载均衡器将连接汇集起来的一项功能，连接处理的处理负荷就其单体来说是非常小的，然而在大规模网站它们就会积少成多，使连接处理本身就成为巨大的负荷。于是，人们利用负载均衡器在前端对客户端连接进行终结处理。这时候，负载均衡器还会另建一个不同于服务器的连接，通过该连接发送请求。这样，服务器只需要保持与负载均衡器之间的连接即可，负荷得以大大减轻。

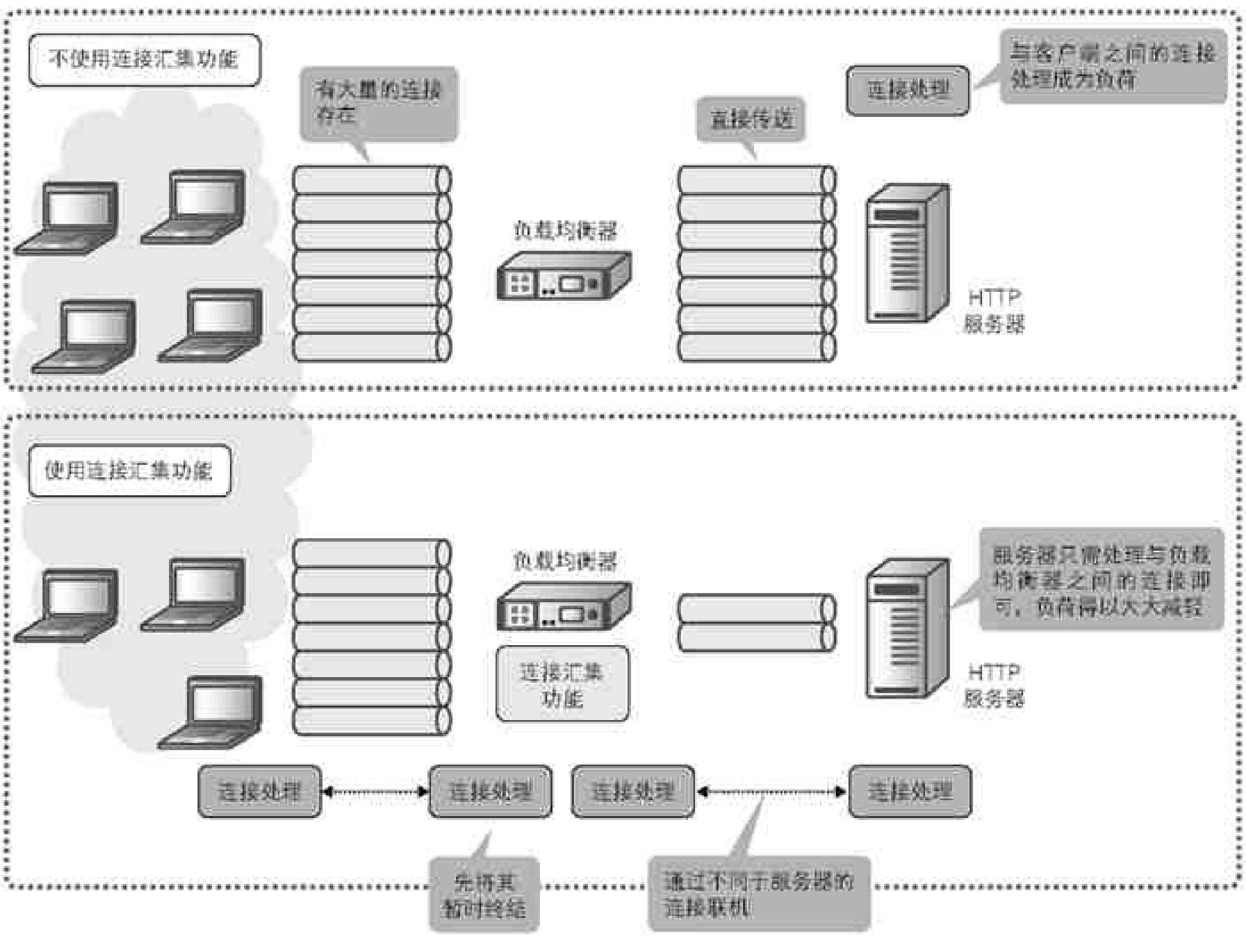


图 **3.1.61**　通过连接汇集功能减轻连接处理的负荷

3.2 从会话层到应用层的技术

前面我们已经学习了物理层到传输层各层的知识，接下来要学习从会话层到应用层的内容了。这些层和从应用程序中衍生出来的应用层协议有着休戚与共的关系，本书将从网络这个角度逐一讲解比较常用的应用层协议。此外，本节并不涉及用于运行管理的协议和用于冗余配置的协议，这两种协议将在第四章和第五章中详细说明。

3.2.1 HTTP 支撑着互联网

在诸多应用协议当中，人们最熟悉的应该就是 HTTP（Hypertext Transfer Protocol，超文本传输协议）。离开这个协议互联网就无从谈起，可以这么说，是 HTTP 让互联网获得了爆发性的成长和席卷性的普及。

HTTP 原本是一种用于传输文本数据的协议，然而如今它早已突破最初的定义范畴，在收发文件和实时交换消息等方面也发挥着巨大的作用。

3.2.1.1 HTTP/1.0 和 HTTP/1.1 的 TCP 连接用法大相径庭

从网络这个角度来看，HTTP 最关键的地方在于它的版本。

当前有两个比较常用的 HTTP 版本，一个是 HTTP/1.0，另一个是 HTTP/1.1。使用哪一个版本取决于我们对浏览器和服务器的设置。最近的浏览器和服务器均默认使用 HTTP/1.1 版本，这使得 HTTP/1.1 成为了主流。

这两个版本的 TCP 连接用法很不一样，极大地影响着服务器的负荷程度以及我们应该选用的设备规格，下面就来详细介绍一下。

HTTP/1.0 为每个请求建立连接

我们可以在 RFC1945 中看到公开的 HTTP/1.0 规格。HTTP/1.0 为每个请求建立 TCP 连接，请求回应结束之后再将其断开，如此反复。举个例子，如果某个网站由 10 条内容构成，HTTP/1.0 就会建立 10 个连接，将每条内容下载之后再断开连接，如此重复多次。当客户端只有唯一的

一个时，这种处理并不会带来很大的负荷，但是当客户端多达 1000 个的时候，情况就大不一样了。积水成渊、聚沙成塔，这样下去最终会给服务器带来巨大的处理负荷。除此以外，我们还需要连接更多的防火墙和负载均衡器，作业量只会有增无减。

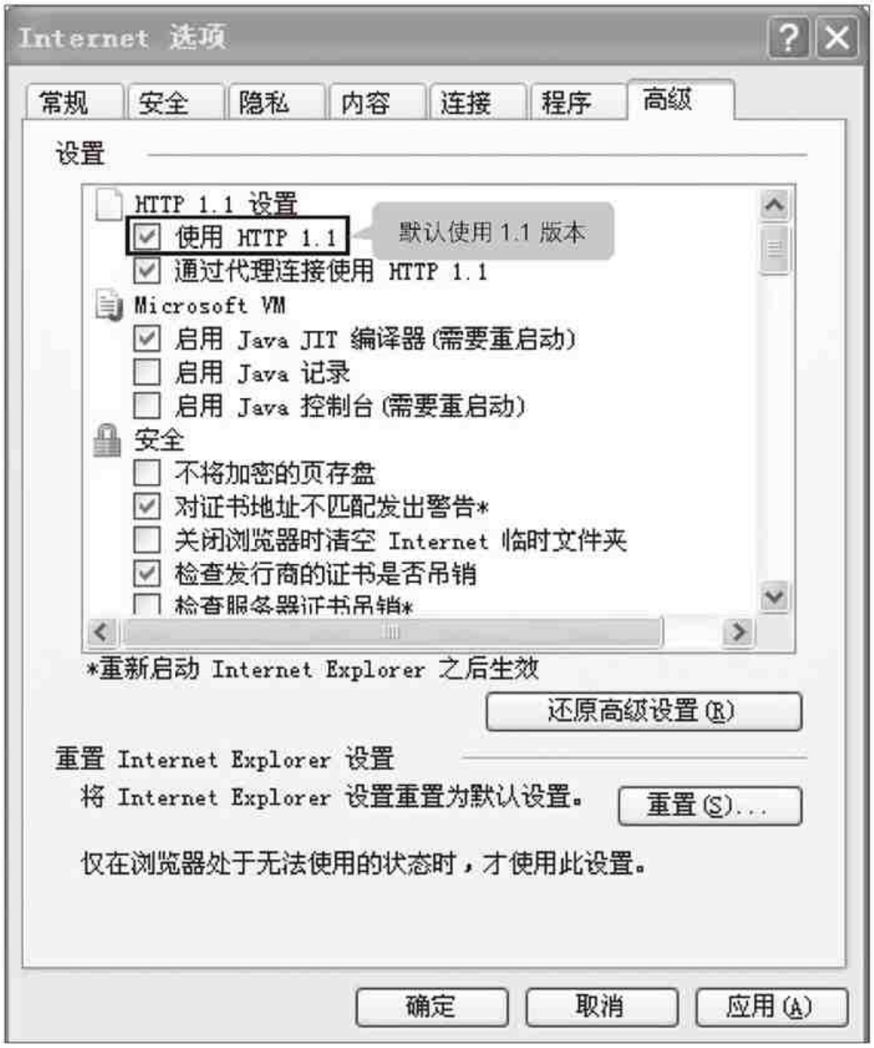


图 **3.2.1**　最近的 **HTTP** 设置均默认使用 **HTTP/1.1** 版本

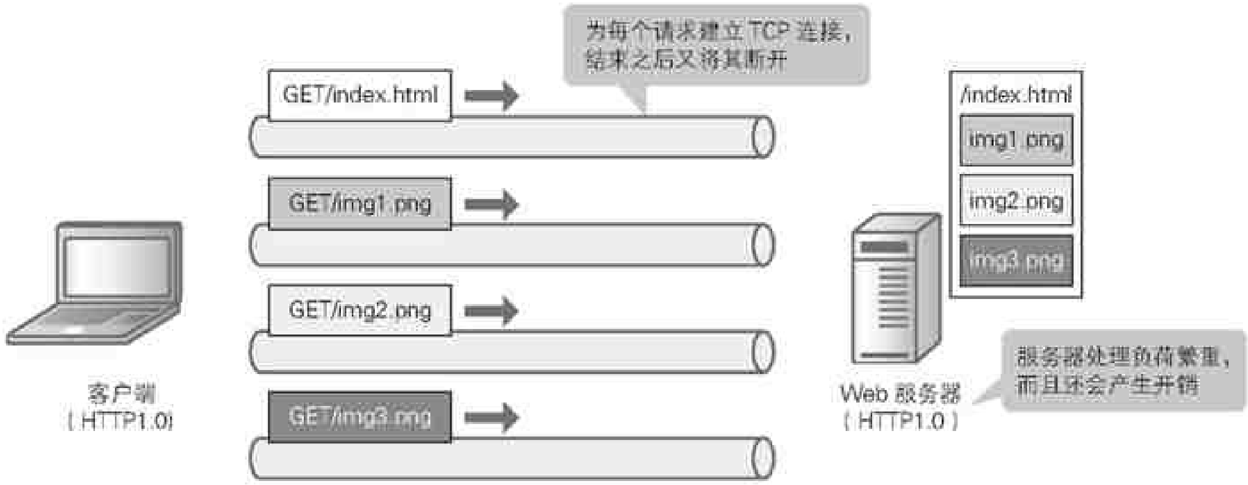


图 **3.2.2　HTTP/1.0** 反复执行建立连接和断开连接的作业

HTTP/1.1 将连接用到极致

我们可以在 RFC2616 中看到公开的 HTTP/1.1 规格。HTTP/1.1 是 HTTP/1.0 的升级版，里面有很多设计都是为了提高 HTTP 效率而开发出来的，其中有一个叫作持久连接，也称作 HTTP Keep- Alive。持久连接在 HTTP/1.0 中还只不过是一项扩展功能而已，但在 HTTP/1.1 中就升级成了标准功能。持久连接是将建好的连接用到极致的一种功能，它会先建立一个 TCP 连接，然后在这个连接上发送多个 HTTP 请求。HTTP/1.0 中的 TCP 处理是建立连接之后又要断开连接，HTTP/1.1 则摒弃了这一规则，大大减轻了服务器的处理负荷。与此同时，还避免了连接更多的防火墙和负载均衡器。

可以同时建立的 TCP 连接数（最大连接数量）取决于浏览器及其版本。如果使用 Internet Explorer 浏览器，按照 RFC 的推荐，7 以前版本的 TCP 连接数为 2，8 以后版本的 TCP 连接数则升为 6。如果使用火狐浏览器，则 TCP 连接数为 6。

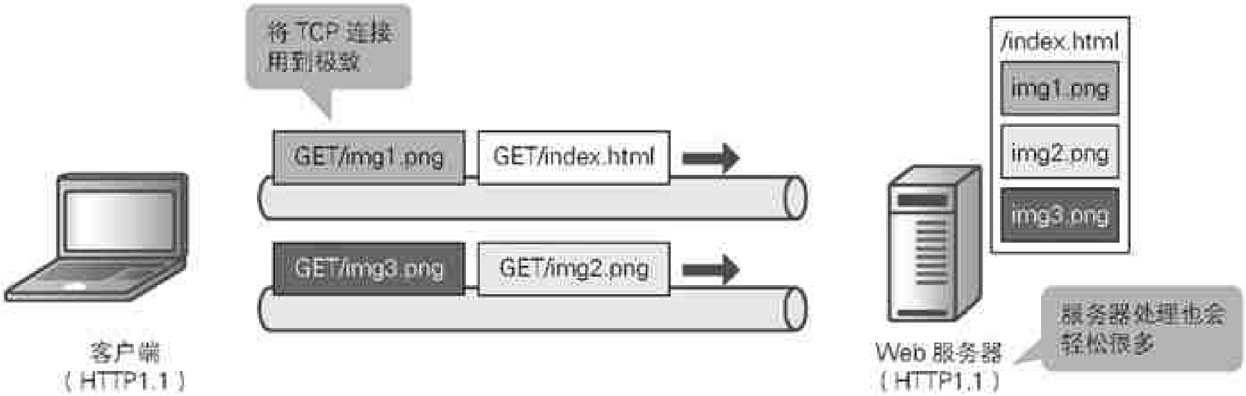


图 **3.2.3　HTTP/1.1** 将连接用到极致

3.2.1.2　HTTP 因请求和响应而得以成立

HTTP 是一种客户端和服务器之间的交互协议，因为有了客户端（Web 浏览器）的 HTTP 请求和服务器的 HTTP 响应，HTTP 才得以成立。客户端向服务器发出 HTTP 请求，例如要求对方将某某文件发过来或者通知对方自己要将某某文件发过去等，服务器收到请求后会进行一定的处理，并将处理结果用 HTTP 响应返回给客户端。

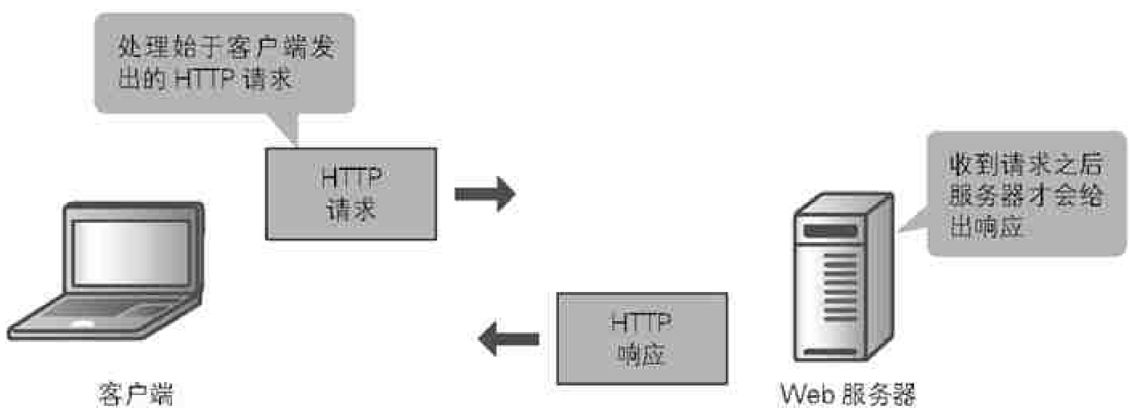


图 **3.2.4　HTTP** 是一种客户端和服务器之间的交互协议

HTTP 消息由消息头和消息体构成

在 HTTP 中交换的 HTTP 消息由两个部分构成，分别为消息头和消息体。

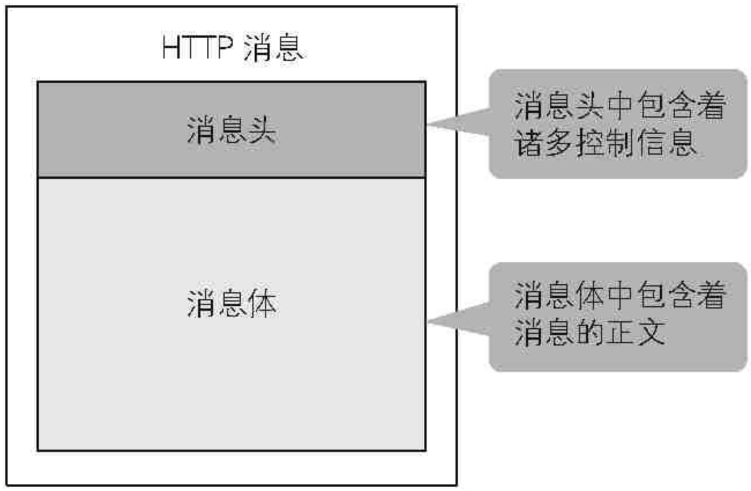


图 **3.2.5　HTTP** 消息由消息头和消息体构成

在这二者当中，对网络来说重要的是消息头。消息头中包含着 HTTP 交互的相关控制信息，这些信息以 HTTP 报头的形式分成几行来记录。客户端和服务器各自根据这些信息去进行压缩处理、持久连接或者识别文件种类。下面，本书就介绍几种常见的 HTTP 报头。

- **Connection** 头 **/Keep-Alive** 头

  Connection 头是用来管理持久连接的 HTTP 报头。Web 浏览器通过 Connection:Keep-Alive 头通知服务器“我支持持久连接哦”。对此服务器会通过 Connection 头返回一个回复，同时还会通过 Keep-Alive 头告诉对方持久连接的相关信息（如时效值和最大请求数等）。

  有些由持久连接建立的连接因为设有 Connection:close 头，可以在时效到来之前就关闭。

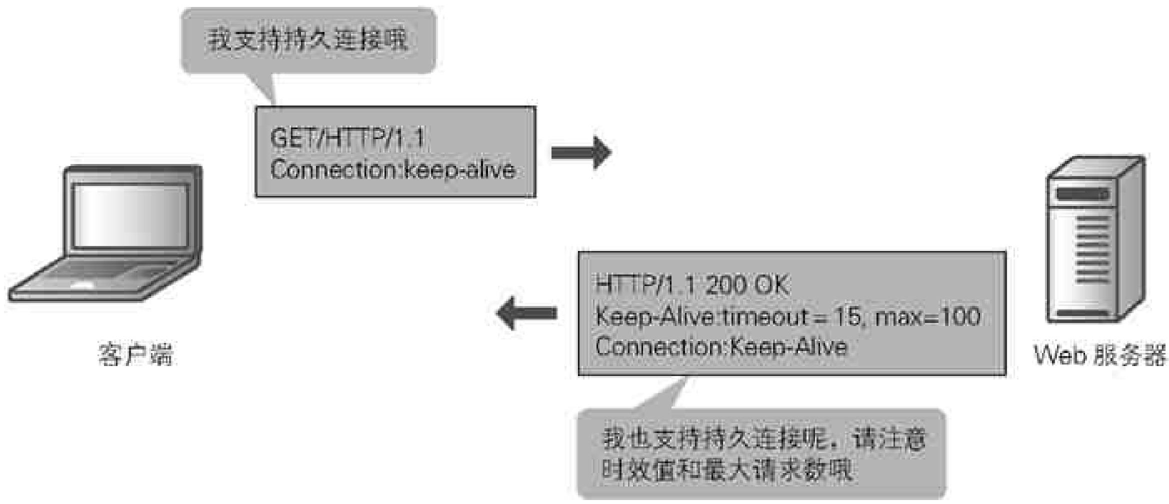


图 **3.2.6**　通过 **Connection** 头管理持久连接

- **Accept-Encoding** 头 **/Content-Encoding** 头

 Accept-Encoding 头和 Content-Encoding 头是用来管理 HTTP 压缩的 HTTP 报头。Web 浏览器通过 Accept-Encoding 头通知服务器"我支持这种方式的压缩哦"，对此服务器会在压缩消息体之后，通过 Content-Encoding 头返回一个"我已经用某某方式压缩好了哦"的回复。

 负载均衡器利用这两种头以及表示文件种类的 Content-type 头对 HTTP 施以高效的压缩。

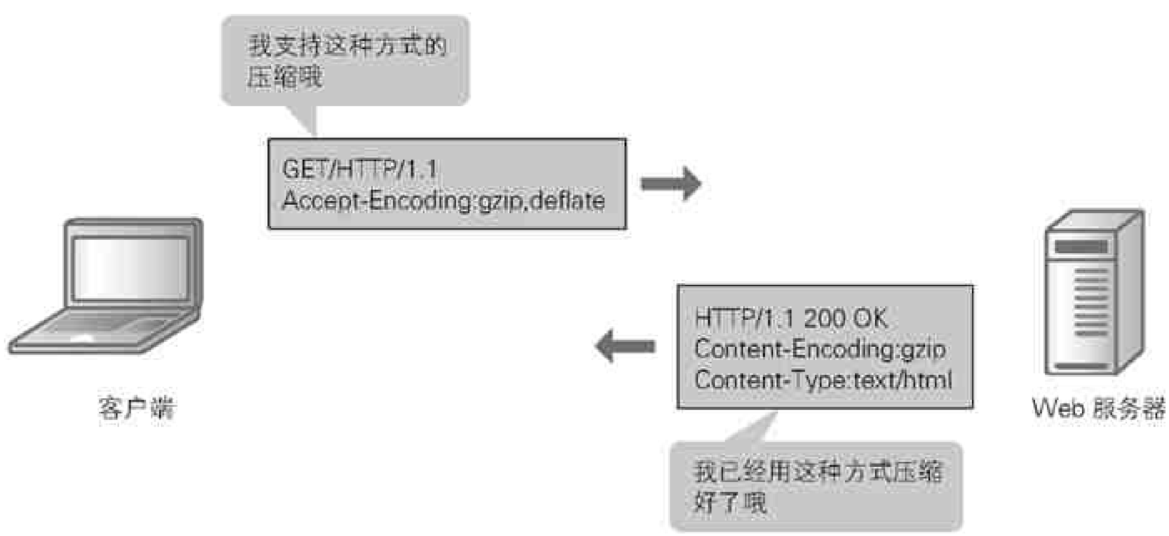


图 **3.2.7**　通过 **Accept-Encoding/Content-Encoding** 头管理 **HTTP**

压缩

- **User-Agent** 头

Web 浏览器将其自身的信息以 User-Agent 头的形式通知给服务器。当你使用低版本的 Web 浏览器时，偶尔会跳出“不支持此浏览器”的警告提示吧？这种警告提示就是根据 User-Agent 头的信息判断出来的。

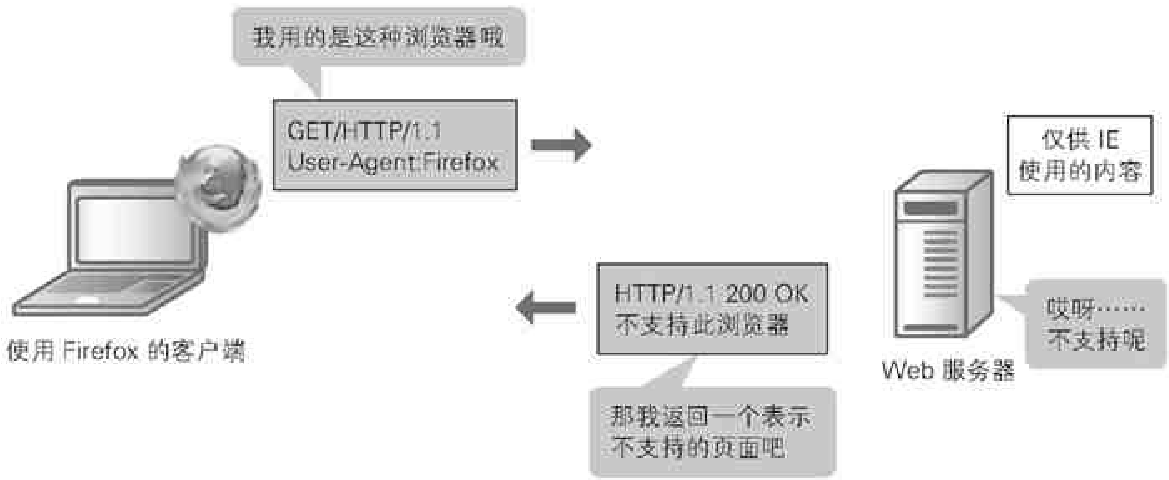


图 **3.2.8**　通过 **User-Agent** 头发送 **Web** 浏览器的信息

※ 事实上 User-Agent 中记载的信息要比此图复杂得多。为了浅显易懂，图中只标出了一些简单的信息。

- **Cookie** 头

Cookie 指的是通过与 HTTP 服务器之间的通信，将特定信息暂时保存到浏览器中的一种机制，同时也指保存这些信息的文件，在 Web 浏览器上的每个 FQDN（Fully Qualified Domain Name，完全合格域名）都拥有相应的 Cookie。大家一定有过下面这样的经历吧，虽然并没有输入自己的用户名和密码，却能够登录购物网站或社交网站，这就是 Cookie 立下的功劳。我们首次输入用户名和密码并在客户端登录成功之后，服务器会发行一个会话 ID 并通过 Set-Cookie 头给出响应。由于后续发出请求时都会在 Cookie 头中加入会话 ID，于是自动登录就得以实现了。

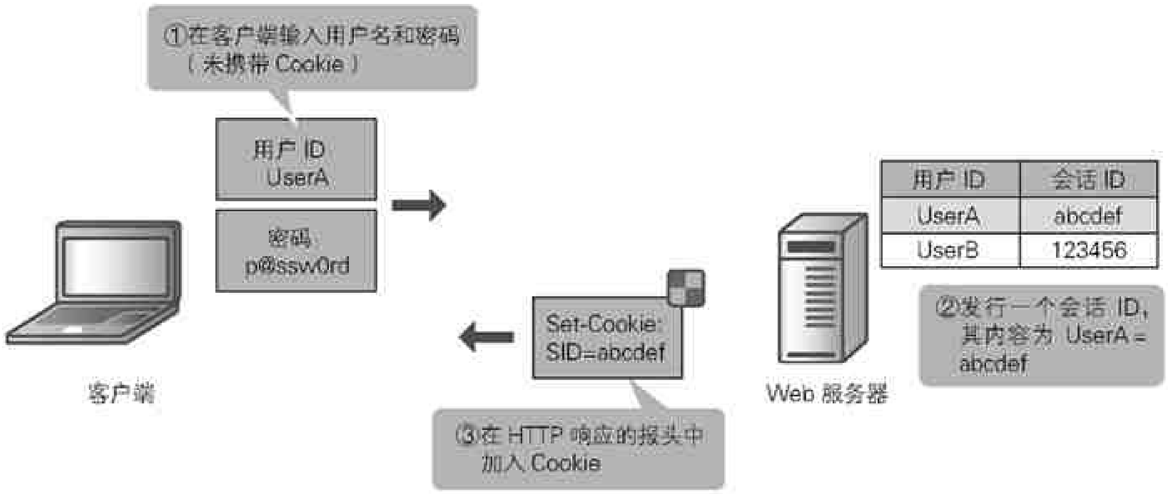


图 **3.2.9**　服务器会发行一个会话 **ID** 并通过 **Cookie** 去交付

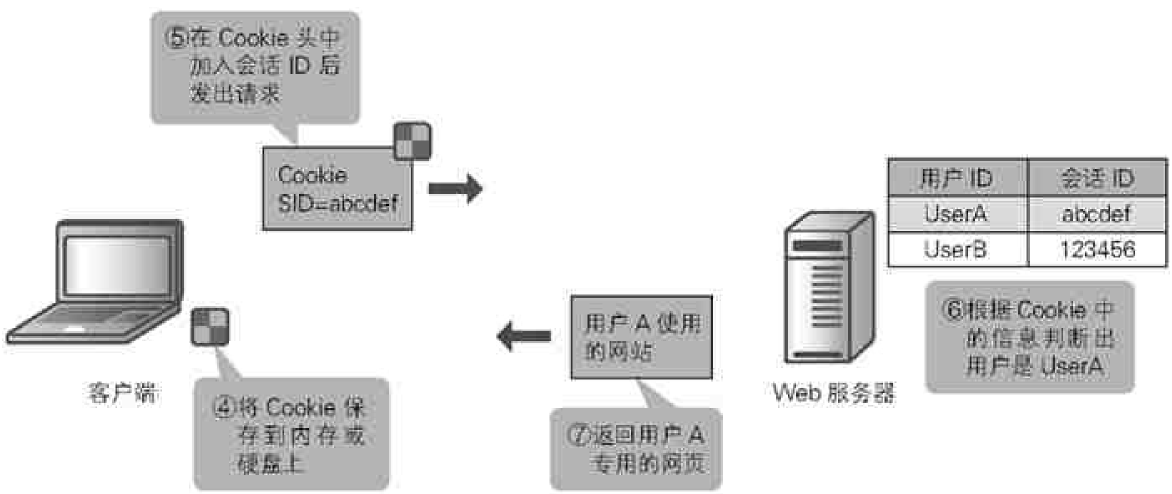


图 **3.2.10**　在 **Cookie** 头中加入会话 **ID** 后发出请求

负载均衡器的 Cookie 会话保持（Insert 模式）利用的就是这种 Cookie 机制。负载均衡器将首次响应时获得的、含服务器信息在内的 Cookie 通过 Set-Cookie 头发送出去，由于后续发出的请求均含有 Cookie 头，信息就能持续不断地发往同一台服务器。

可通过多种方法发出 **HTTP** 请求

客户端用请求消息头中最前面记述的请求行传达 HTTP 请求的内容。请求行由三个要素构成，分别是方法、URI（Uniform Resource Identifier，统一资源标识符）和 HTTP 版本。方法指 HTTP 请求的种类，URI 指请求的地点（/index.html、http:// ～ 等）。客户端向 URI 要求相应的方

法，服务器将处理结果返回。



图 **3.2.11**　通过请求行传达 **HTTP** 请求的内容

用于请求行的方法只有寥寥几种，非常简单。下表中列举了一些常用的方法以供大家参考。

表 **3.2.1**　方法表示请求的种类

方法	内容
OPTIONS	由服务器查询支持的方法和选项
GET	从服务器获取数据
HEAD	仅获取消息头
POST	将数据传送给服务器
PUT	将本地文件传送给服务器

DELETE	删除文件
TRACE	确认通往服务器的路径
CONNECT	向代理服务器要求隧道穿越

用状态码传达状态

服务器收到 HTTP 请求之后，用响应消息头中最前面记述的状态行返回处理结果。状态行由三个要素构成，分别是状态码、状态码说明和 HTTP 版本。

这三个要素中最重要的是状态码。状态码是一个三位数的编号，表示处理结果如何。每个状态码都有着不同的含义，例如，Web 服务器运行正常时会返回“200 OK”，请求内容并不存在时会返回“404 Not Found”。

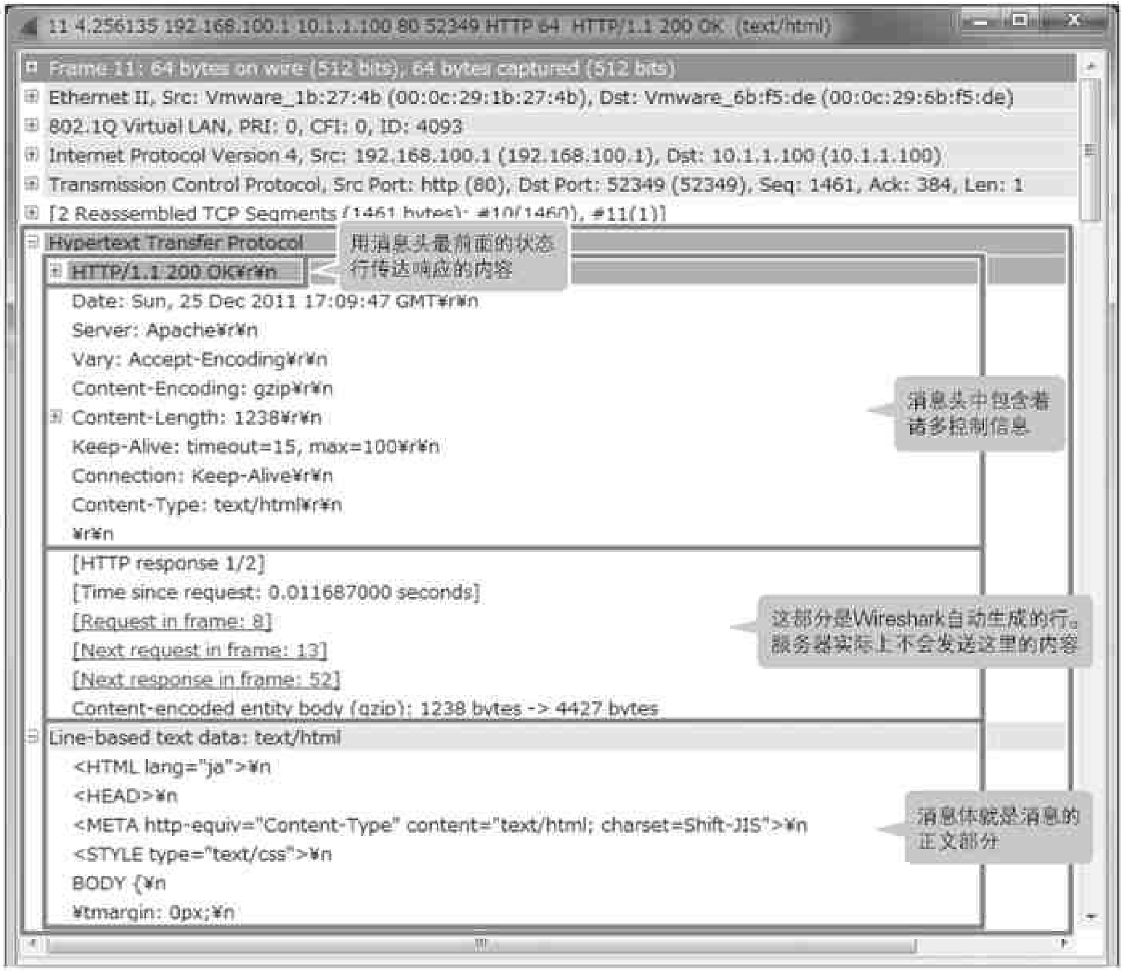


图 **3.2.12**　通过状态行返回处理结果

表 **3.2.2**　用状态码传达服务器的状态

状态码	概要	内容
1xx	Informational（信息）	收到请求，正在处理
2xx	Successful（成功）	已成功处理
3xx	Redirection（重定向）	要想完成处理，还需进行进一步的操作
4xx	Client Error（客户端错误）	请求的语法不对或者是服务器未能接收

5xx	Server Error（服务器错误）	请求已收到但无法处理

3.2.2　用 SSL 保护数据

SSL（Secure Socket Layer，安全套阶层）是一种保护应用数据不受网络不法侵害的协议。人们在网络上交换地址、电话号码、信用卡号、企业机密等各种信息，网络上的恶意破坏者则始终都在伺机窥探，企图窃取这些重要的信息。

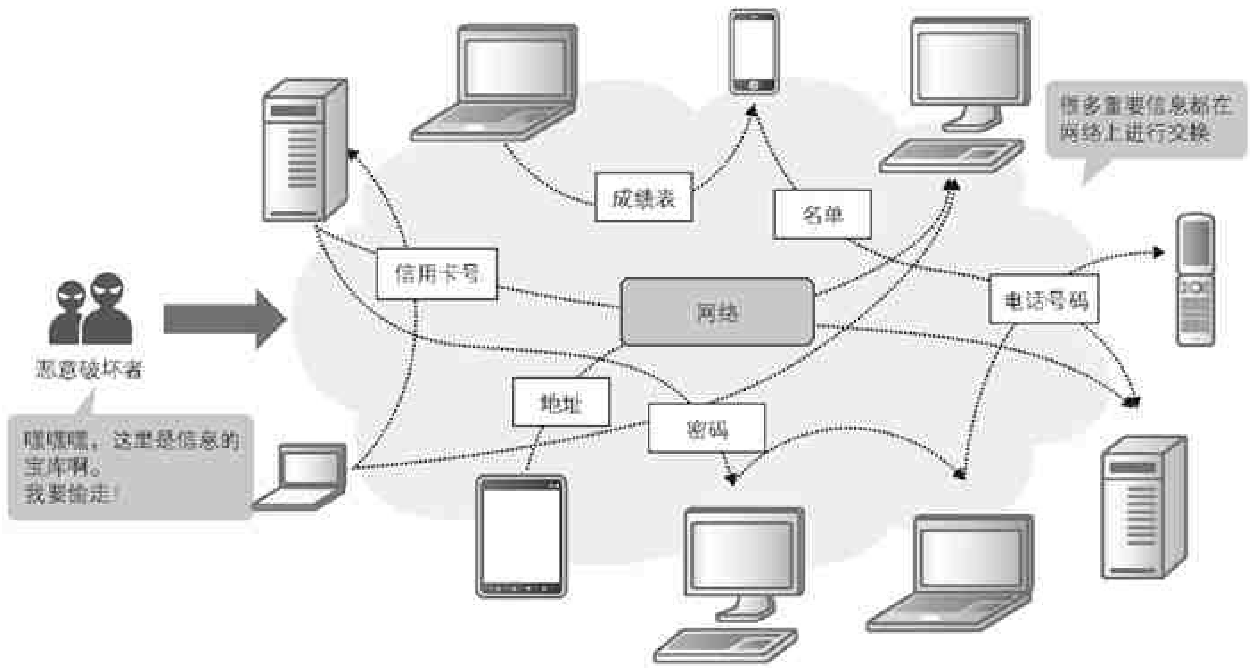


图 **3.2.13**　人们在网络上交换各种各样的信息

SSL 是为了保护数据不受恶意破坏者侵害而出现的。人们用 SSL 将客户端和服务器之间的数据加密并进行认证，以此来保护重要的数据。大家在上网时都可以看到，Web 浏览器的网址显示为“https:// ～ ”的形式，后面有一把上锁的标记。这表示该网页已被 SSL 加密，数据是非常安全的，不会外泄。HTTPS 是 HTTP over SSL 的简称，是指对 HTTP 做了 SSL 处理。



图 **3.2.14**　可通过 **Web** 浏览器的显示内容判断该网页是否已做了 **SSL** 处理

3.2.2.1　防止窃听、篡改和冒充

互联网让我们能够搜索到自己需要的信息，能够快速地购买商品，是一个非常方便的世界。然而与此同时，互联网也是一个充满了危险的世界，其中的三大危害就是窃听、篡改和冒充。SSL 能够保护数据免受侵害，下面我们就来看一看，它分别都运用了哪些技术去抵御这些侵害。

用加密技术防止窃听

窃听应该是最容易理解的一种侵害。重要数据如果不加任何处理就公开交互，那么即使不是恶意破坏者的普通人也会对其中的内容产生好奇吧。SSL 将客户端和服务器之间的通信加密，这样，即使外人窃听到这些信息，也无从获悉其中的具体内容。

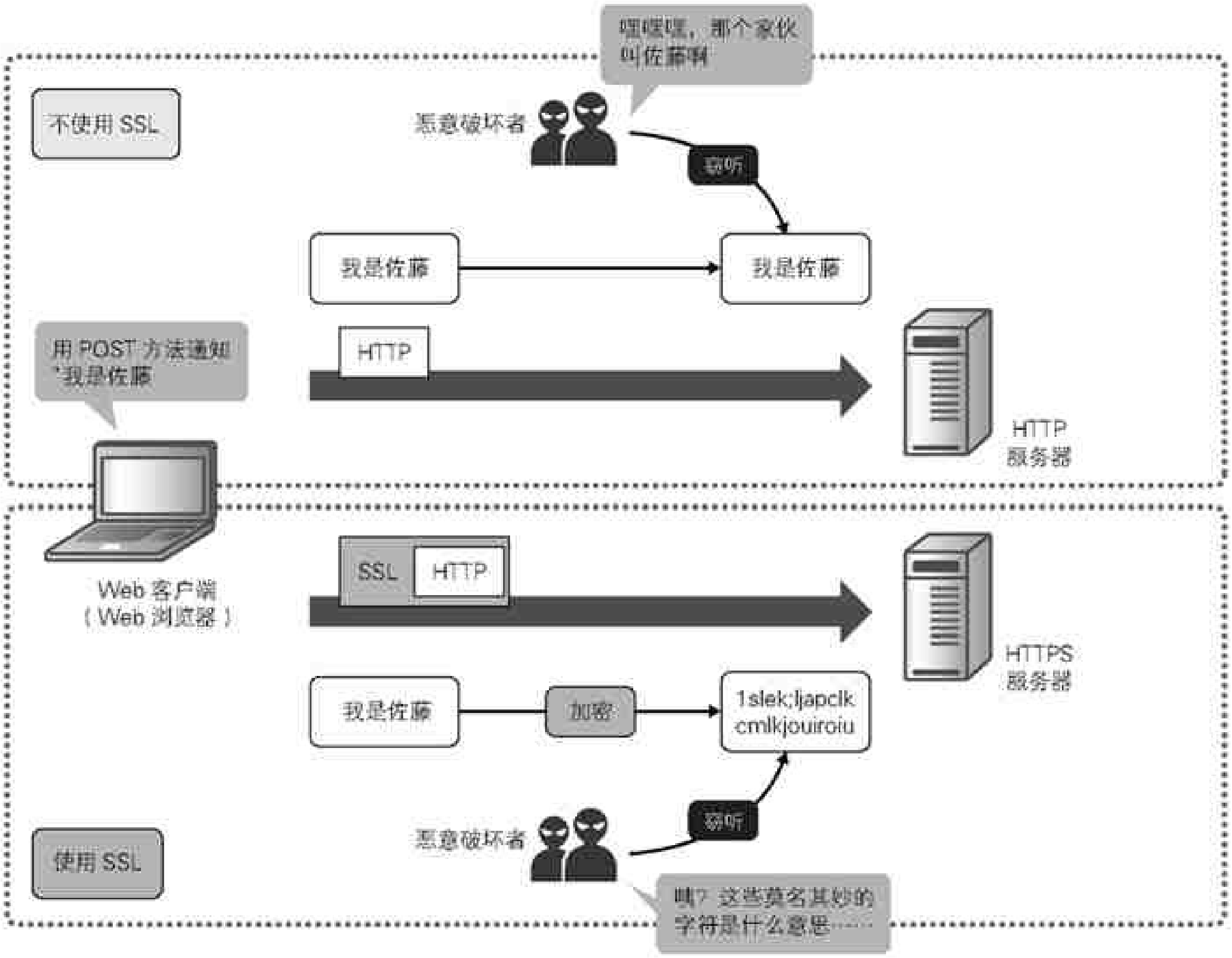


图 **3.2.15**　用加密技术防止窃听

用消息摘要技术检测数据是否被篡改过

篡改是指数据在传输途中被擅自改写的一种侵害。假设我们在网上订购了商品，但是如果送货地址被恶意破坏者改成了他自己的地址，我们就无法收到该商品了。SSL 将根据数据计算出的消息摘要（MD 值）和数据本身一起发送，以此来检查数据是否被篡改过。节点收到它们之后，将根据数据计算出的消息摘要和添加的消息摘要进行比较，看二者是否一致。由于是对同一数据进行同样的计算，如果哈希值一样就说明数据并未被篡改。

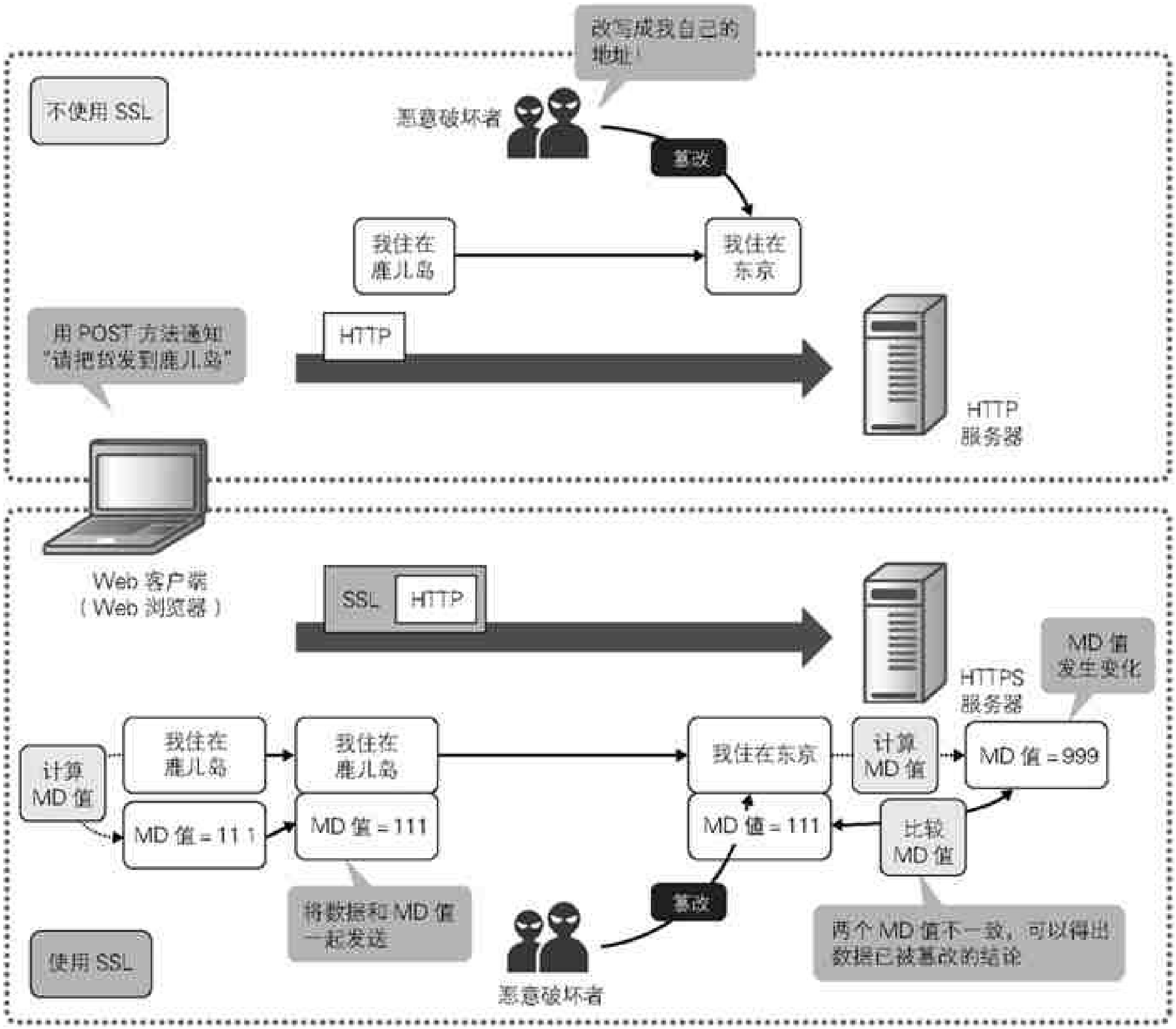


图 **3.2.16**　用消息摘要技术检测数据是否被篡改过

用数字证书技术识破冒充

冒充是假扮成通信对象的一种侵害。数据发送方并不知道对方是否真的是自己想要发送数据的对象，通信可能在不知不觉中就被恶意破坏者劫持，而且这些恶意破坏者还可能冒充本应收到数据的一方。所以，如果我们不采取任何防范措施，就相当于把重要的数据亲自交到坏人的手上。因此，SSL会利用数字证书去确认对方的身份，它的工作原理是，在发送数据之前要求对方提供自身的信息，然后根据对方发过来的数字证书验明正身。

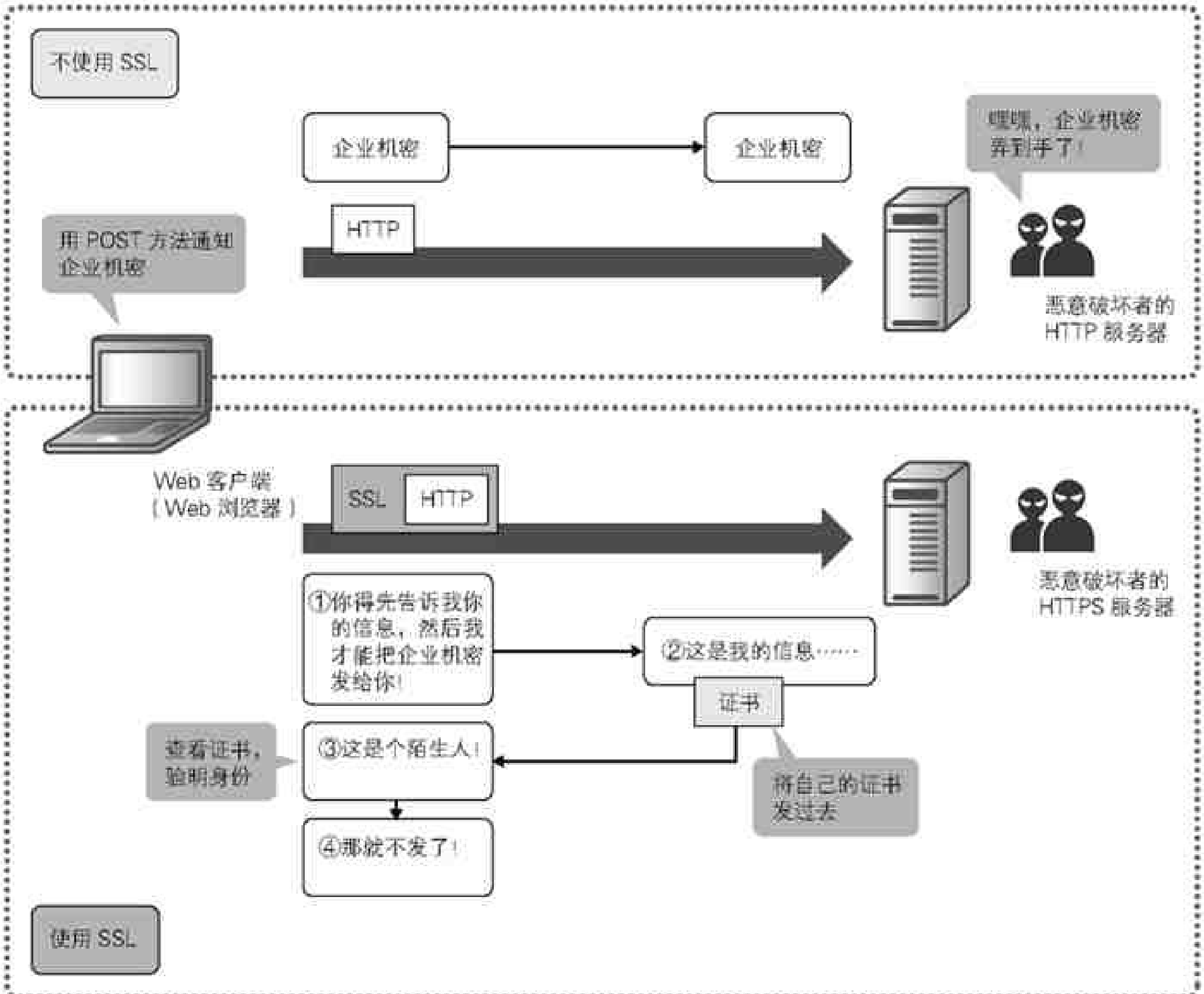


图 **3.2.17**　用数字证书技术识破冒充

3.2.2.2　通过 **SSL** 可以给各种各样的应用程序协议加密

常常有人误以为 SSL 是 HTTP 专用的加密协议，其实并非如此，只不过碰巧 HTTPS 是网络上最为常用的协议，所以容易让人们产生这样的误解而已。SSL 在传输层运作，和应用协议是各自独立工作的。由于 SSL 要用到 TCP，所以它应该是在 TCP 之上应用程序之下，给人感觉是在第 4.5 层中运作。SSL 将 TCP 应用协议视为加密对象，因此，用于文件传输的 FTP 和用于邮件收发的 SMTP 也可以用 SSL 进行加密处理。这时候，这些协议的称呼也会变成“○○ over SSL”的形式，FTP 变成 FTP over SSL，SMTP 则变成 SMTP over SSL。

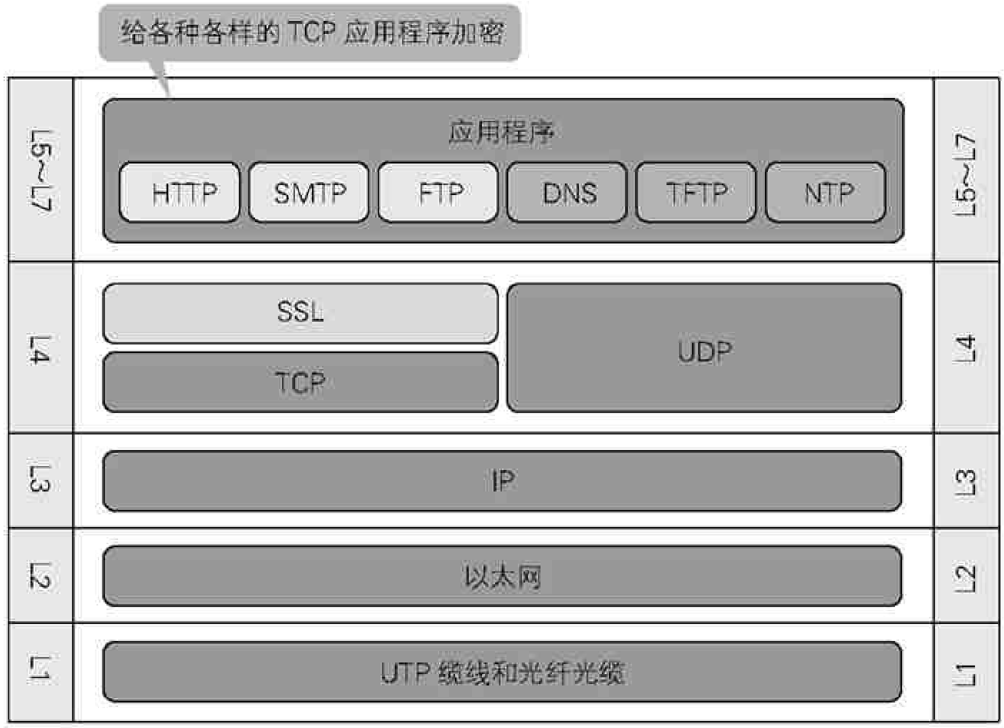


图 **3.2.18**　可通过 **SSL** 给各种各样的 **TCP** 程序加密

3.2.2.3　SSL 使用混合加密方式进行加密

加密技术由加密和解密的双重关系构成。

发信方将需要发送的原文和用于加密的钥匙“加密密钥”装入一个叫作加密算法的数学计算步骤中，然后上锁，将原文转换为加密文（加密处理）。收信方将收到的加密文和用于解密的钥匙“解密密钥”装入一个叫作解密算法的数学计算步骤中，然后开锁，由此取出原文（解密处理）。根据加密密钥和解密密钥的使用方法，网络中的加密技术大致分为共享密钥加密和公钥加密两种方式，下面来分别说明一下。

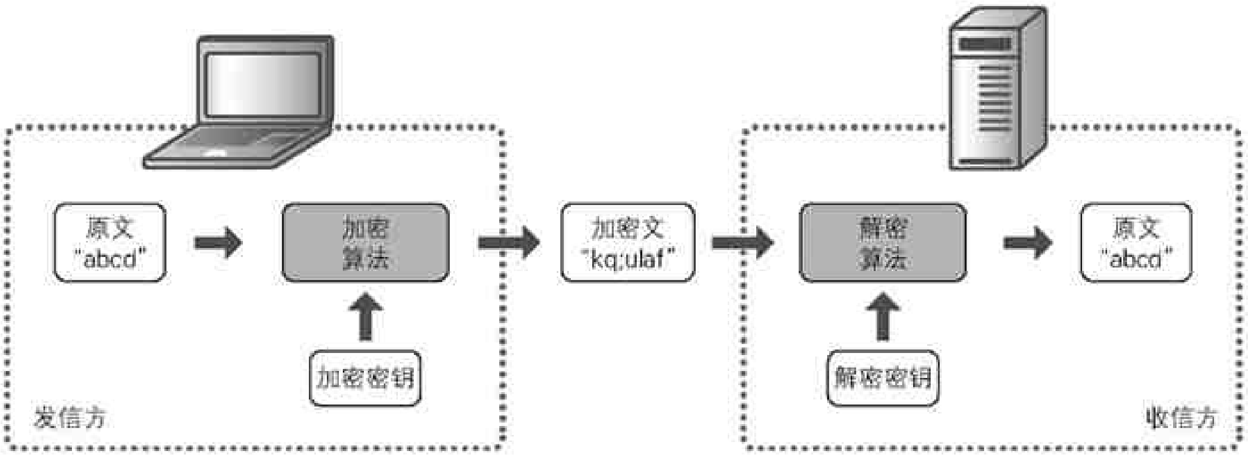


图 **3.2.19**　加密技术由加密和解密的双重关系构成

用共享密钥加密方式可进行高速处理

在共享密钥加密方式中，加密密钥和解密密钥是相同的。由于发信方和收信方对称地使用相同的密钥，所以这种方式又被称作对称密钥加密方式。发信方和收信方预先共享同一把密钥，用加密密钥将数据加密，然后用和加密密钥完全一致的解密密钥为数据解密。DES（Data Encryption Standard，数据加密标准）、3DES（Triple Data Encryption Standard，三重数据加密标准）和 AES（Advanced Encryption Standard，高级加密标准）都属于这一类加密方式。

共享密钥加密方式的优点在于它的处理速度，由于结构简单，加密处理和解密处理都能够高速地完成。缺点则是密钥的传送问题，由于加密密钥和解密密钥相同，万一密钥被恶意破坏者截取，就无法保证数据安全了。所以，我们必须另外考虑如何将通信双方之间共享的密钥安全地传送给对方。

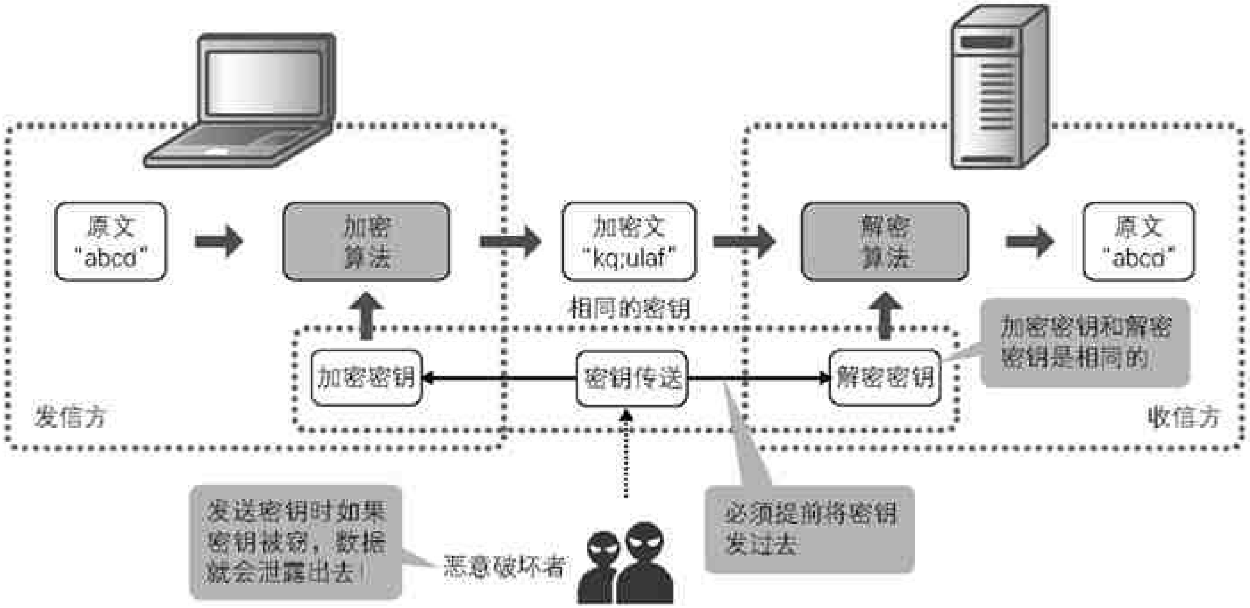


图 **3.2.20**　共享密钥加密方式中的加密密钥和解密密钥是相同的

用公开密钥加密方式能更好地解决密钥传送问题

公开密钥加密方式（公钥加密方式）是加密和解密分别使用不同密钥的加密方式。由于发信方和收信方不对称地使用不同的密钥，所以这种方式又被称为非对称密钥加密方式。RSA 和椭圆曲线加密属于这一类加密方式。

在公钥加密方式背后起着支撑作用的是公钥和秘钥。它们各如其名，公钥指可以公开的密钥，秘钥则指秘密保管的密钥，这两个密钥是成对使用的，所以被称为密钥对。密钥对之间存在着某种数学关系，不能用其中一把计算出另外一把。而且，用公钥加密的加密文必须用秘钥才能解密。

那么，密钥对在公开密钥加密方式中是如何运作的呢？我们来看一下它的处理顺序。

1 →　收信方生成公钥和秘钥（密钥对）。

2 →　收信方将公钥对外发布，同时保管秘钥。

3 →　发信方将公钥加密后发给收信方。

4 → 收信方用秘钥给公钥解密。

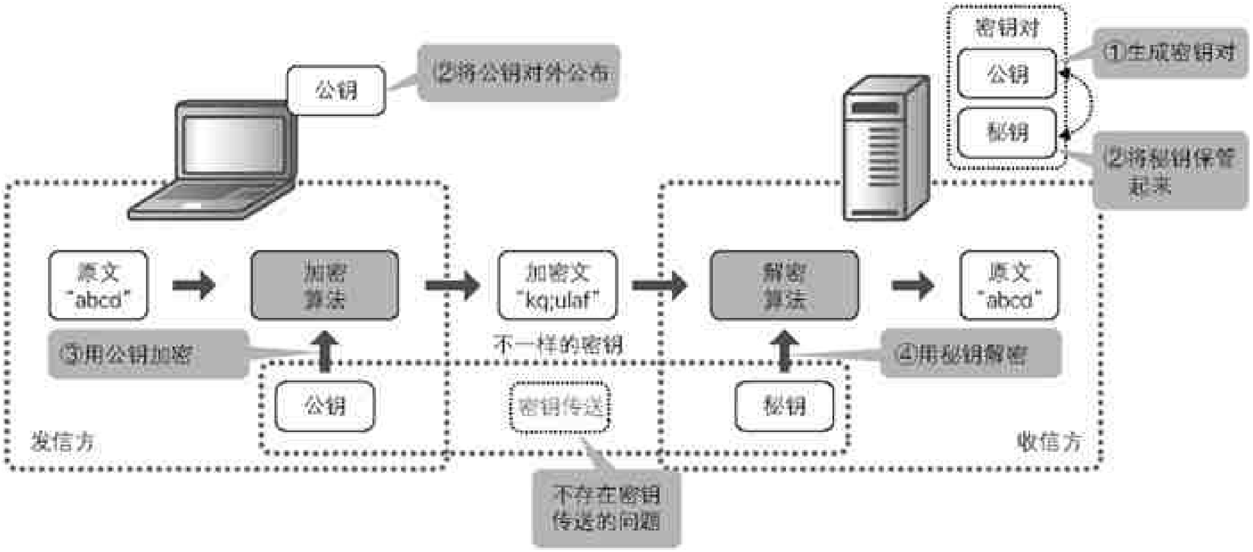


图 **3.2.21**　公钥加密方式使用公钥和秘钥

公钥加密方式的优点在于密钥传送。用于加密的公钥是对外公开的密钥，只有和秘钥在一起才能发挥作用，而且谁也无法由公钥计算出秘钥，因此我们不必担心密钥传送是否安全这个问题。缺点则是处理速度较慢，公钥加密方式的处理比较复杂，加密和解密都需要耗费较长的时间。

混合加密方式能采众家之长

在公钥加密方式和共享密钥加密方式中，一方的优点恰好是另一方的缺点，反之亦然。在这样的背景下，混合加密方式应运而生。SSL 采用的就是混合加密方式，它既能像共享密钥加密方式那样进行高速处理，又能像公钥加密方式那样解决密钥传送的问题，是一种采众家之长的加密方式。

表 **3.2.3**　在共享密钥加密方式和公钥加密方式中，一方的优点恰好是另一方的缺点，反之亦然

比较事项	共享密钥加密方式	公钥加密方式
处理速度	高速	低速

密钥的传送问题	存在	不存在
密钥的管理	每个通信对象都需要管理	只需管理秘钥即可

在混合加密方式中，消息是通过共享密钥加密方式加密的，使用共享密钥加密方式可实现高速处理。与此同时，共享密钥加密方式中使用的密钥是通过公钥加密方式加密的，这样又能够解决密钥传送的问题。

下面我们来看一下实际的工作流程。1 ～ 4 为公钥加密方式，5 ～ 6 为共享密钥加密方式。

1 → 收信方生成公钥和秘钥。

2 → 收信方将公钥对外发布，同时保管秘钥。

3 → 发信方用公钥将共享密钥（在共享密钥加密方式中使用的密钥）加密后发给收信方。

4 → 收信方用秘钥解密后取出共享密钥，从这一刻开始，双方共同拥有给消息加密和解密的密钥。

5 → 发信方用共享密钥给消息加密后发给收信方。

6 → 收信方用共享密钥给收到的消息解密。

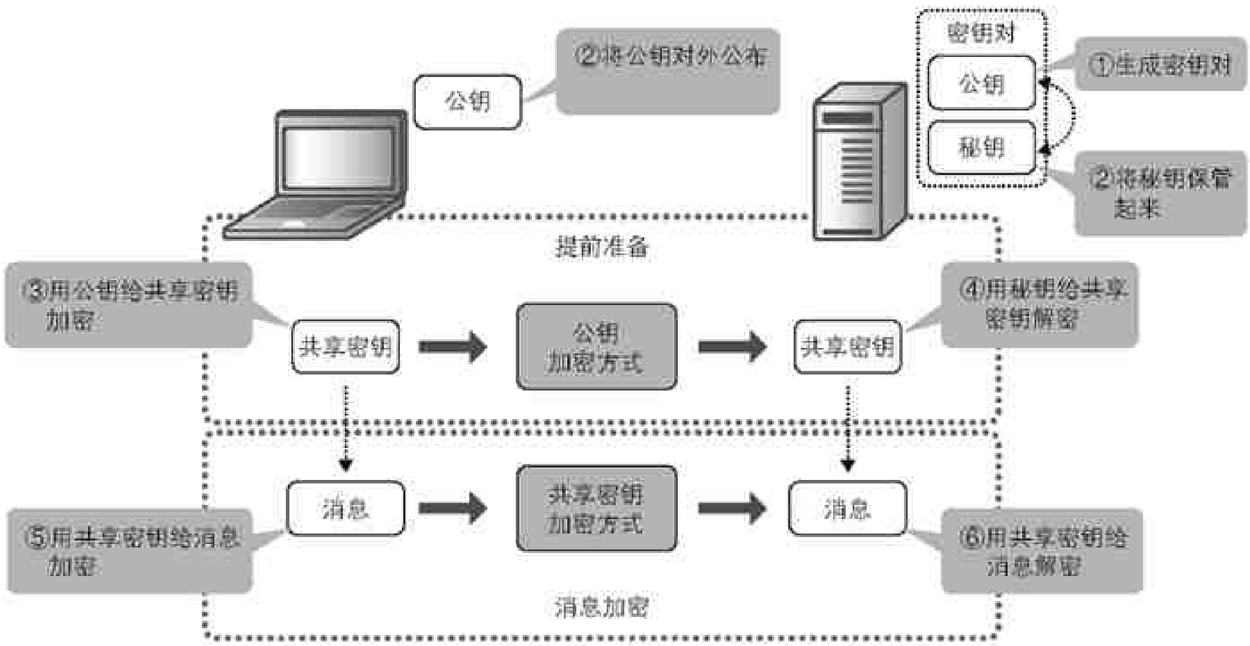


图 **3.2.22**　混合密钥加密方式采众家之长

3.2.2.4　消息摘要是消息的概要

正如其名，消息摘要指的是消息的概要（摘要）。摘要的提取犹如采集数据的“指纹”，所以它又被称作（数字）指纹。另外，由于它像做炸土豆饼（Hashed Potato，谐音为哈希土豆）一样是将数据“切碎”再进行混合加工，所以也被称作哈希（Hash）[7]。

> [7] Hash 此处采用了音译，也可译为“散列”。——译者注

比较消息摘要效率更高

当我们需要查看两个数据是否一致时，直接将二者进行比较是最简单也最直截了当的办法。当然，如果数据比较小，用这种办法也是可行的。但是，如果数据很大会怎么样呢？结论是保管数据可能就需要成倍的空间，比较作业本身也会非常麻烦，于是人们想到了用消息摘要来解决这个问题。消息摘要是由“单向哈希函数”计算出来的值。因为它是数据的摘要，所以比较小，而且无论原来的数据有多大，算出来的消息摘要的大小都是相同的，因此处理起来非常方便。另外，人们无法根据消息摘要复原出完整的数据，也就是说，即使它不幸被窃，从安全角度上说也没有任何问题。

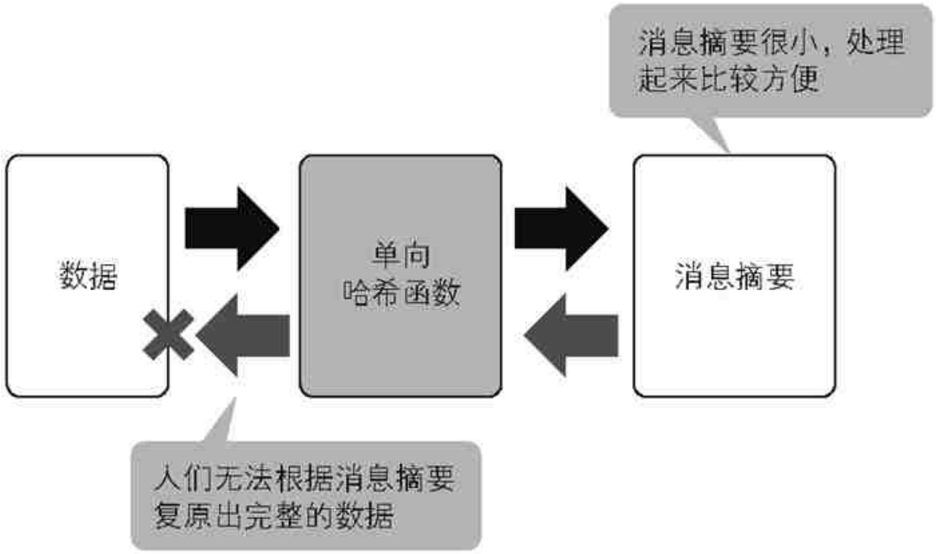


图 **3.2.23**　由单向哈希函数生成消息摘要

比较数据的时候，先是由发信方将数据和消息摘要一起发给收信方，收信方根据数据算出消息摘要之后，将自己算出的消息摘要和收到的消息摘要进行对比。如果数据未被篡改过，那么经过同样的计算得出的消息摘要就应该是一致的。

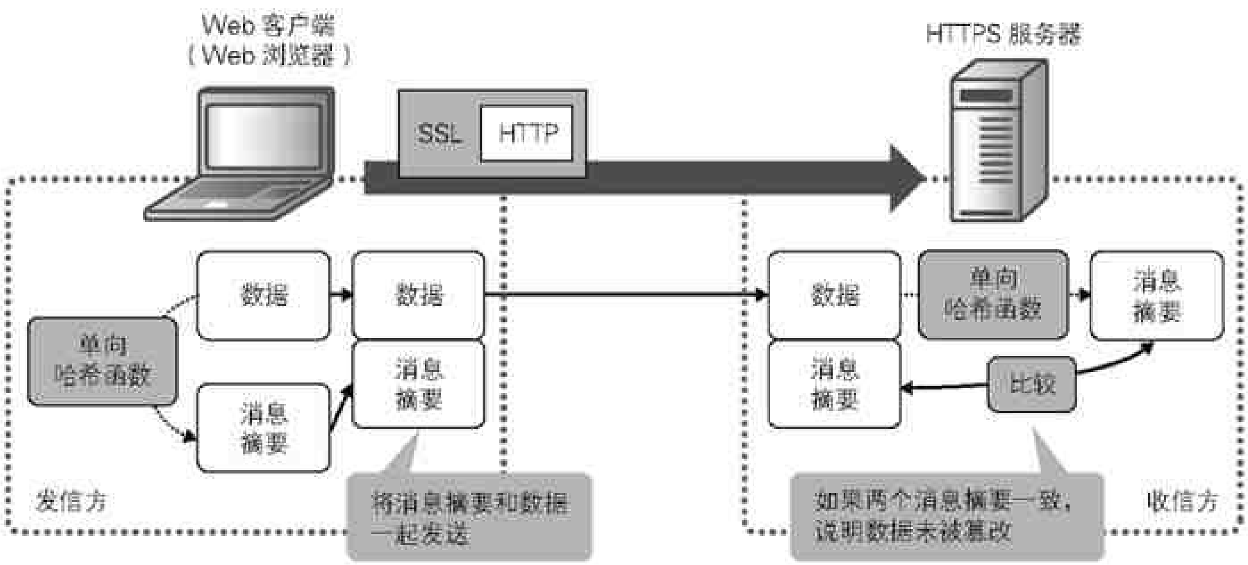


图 **3.2.24**　用消息摘要（**MD**）检测数据是否被篡改过

通过消息摘要进行证书验证

在 SSL 中，消息摘要还用于验证数字证书。无论做了多么周全的加密

处理，如果发送数据的对象是陌生人，那么通信就没有任何意义。在SSL 中，发信方和收信方利用数字证书来证明自己和对方的身份。这样的话，有一点就非常重要了，那就是即使某一方大声呼喊“我是 A 哦！”也不足以令人信服。口说无凭，这个人到底是不是 A 我们无从知道，也许只是某个人 B 在大叫“我是 A 哦！”而已。因此在通信中，SSL 是采用第三方认证来解决这个问题的。具体说来，是让可以信任的第三方——CA（Certificate Authority，证书授权）机构以数字签名的形式对“A 是 A”这个事实给予认可。然后在该数字签名中使用消息摘要。

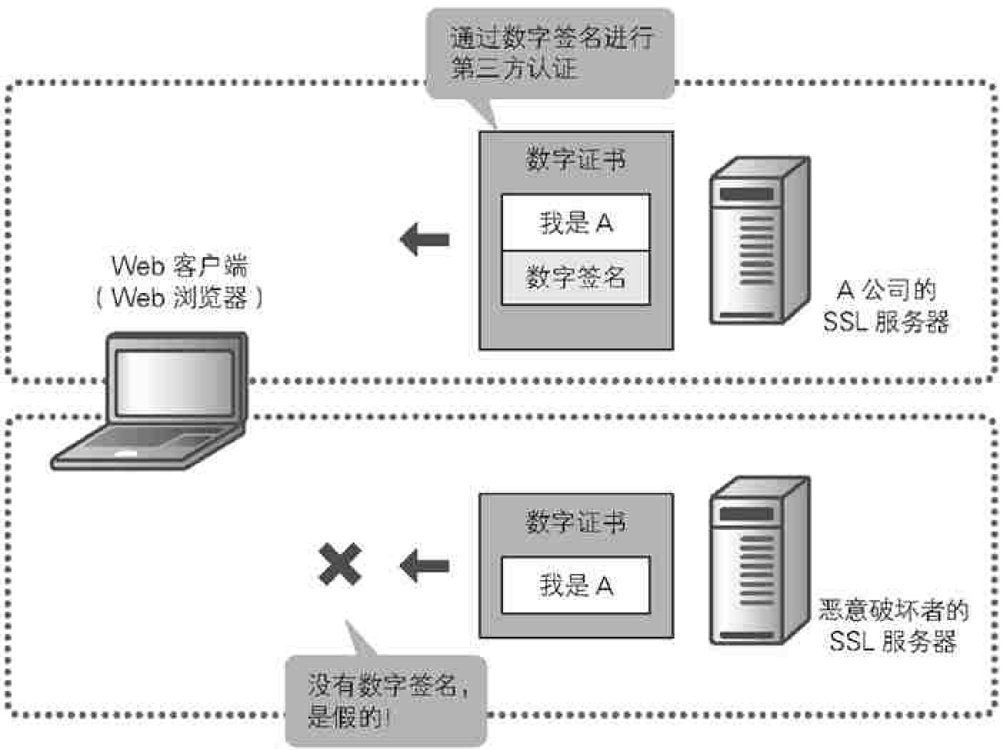


图 **3.2.25**　通过数字签名进行第三方认证

数字证书由签名前证书、数字签名算法和数字签名这三个部分构成。签名前证书是服务器和服务器持有人的信息，表示服务器网址的公用名称、有效期限和公钥也包含在其中。数字签名算法中包含单向哈希函数。将签名前证书经过哈希处理得到的消息摘要用 CA 机构的秘钥进行加密，就得到了数字签名。

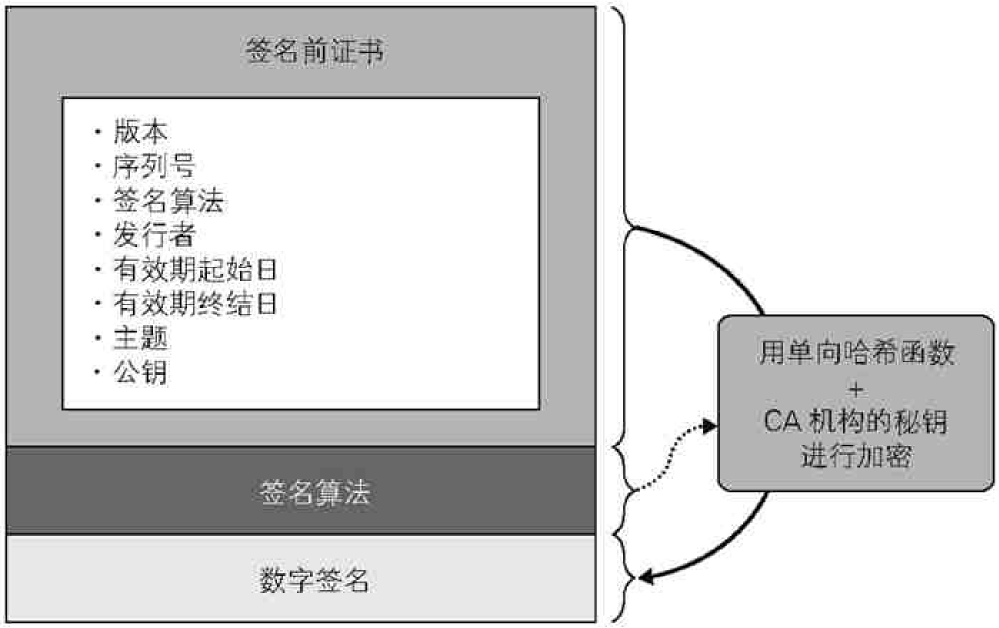


图 **3.2.26**　数字证书由签名前证书、数字签名算法和数字签名三个部分构成

收信方收到数字证书之后，用 CA 机构的公钥（路由证书）将数字签名解密，然后将其和签名前证书对比。如果二者一致就说明证书未被篡改，也就是说发来信息的服务器的确不是由恶意破坏者冒充的。

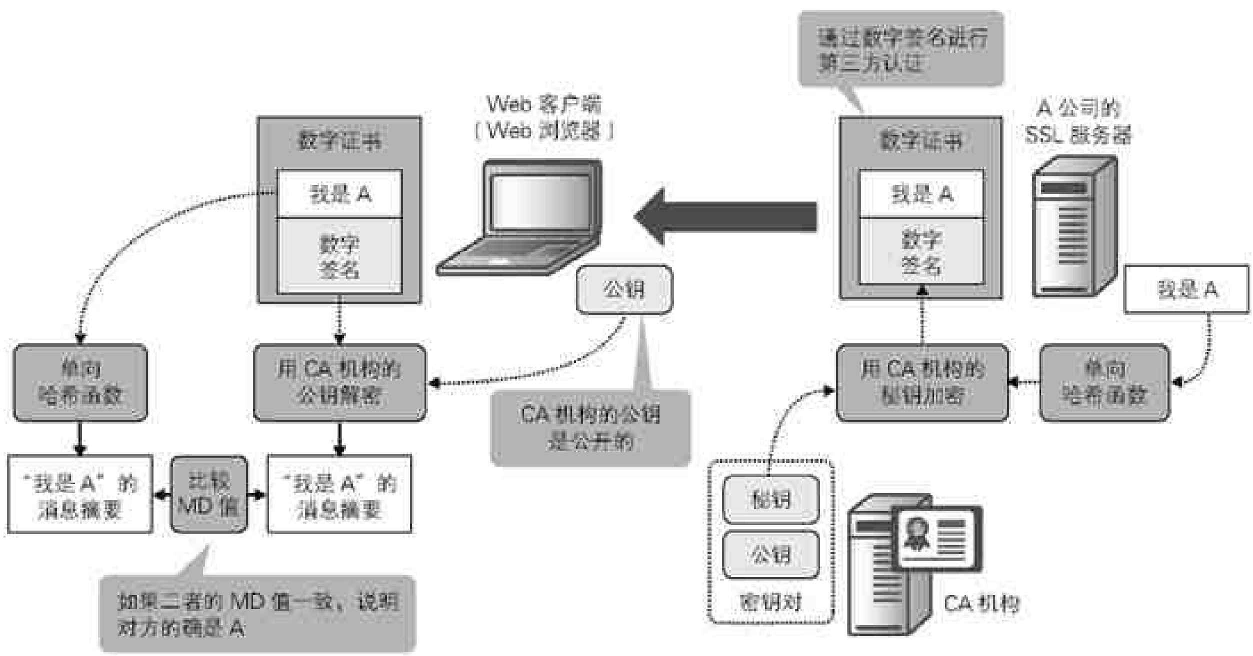


图 **3.2.27**　验证证书

补充一下，如果数字证书验证失败，则会跳出警告提示。

3.2.2.5　SSL 中执行着大量的处理

SSL 中包含着诸多技术组合，是一种综合性的加密协议。为了让这些技术彼此组合并相辅相成，最终实现成功连网，需要执行大量的处理。下面，我们以某台 SSL 服务器要在互联网上公开为前提，逐一梳理各项处理的具体内容。

准备服务器证书并将其安装到服务器上

在互联网上公开 SSL 服务器，并不意味着准备好一台已启动 SSL 服务的服务器就足够了。将一台服务器作为 SSL 服务器公开之前我们需要做一些准备工作，例如准备好证书、向 CA 机构提交申请，等等。大致说来可分为以下四个步骤。

1 →　通过 SSL 服务器 [8] 生成秘钥。秘钥是不可以对外公开的密钥，一定要细心保管，避免遗失。

> [8] 如果是通过负载均衡器进行 SSL 加速，则也由负载均衡器生成密钥对。

2 →　用**1 →** 中生成的秘钥生成 CSR（Certificate Signing Request，证书签名请求），将其发给 CA 机构。CSR 是一种为获得服务器证书而提交给 CA 机构的随机字符串，由签名前证书的信息构成，生成时会将相关的信息一一写入。生成 CSR 时需要用到的信息统称为区别名称，表 3.2.4 中列出了其中所包含的内容。

表 **3.2.4**　写入区别名称，生成 **CSR**

所需信息	内容	例
公用名称	Web 服务器的网址（FQDN）	www.local.com
组织名称	该网站运营组织的正式英语名称	Local Japan K.K
部门名称	该网站的运营部门及其下属的名称	Information Security Section

城镇名称	该网站运营组织的地址	Kirishima
省、直辖市名称	该网站运营组织的地址	Kagoshima
国家代码	国家代码	JP

不同的 CA 机构会要求不同的申请提交信息和公钥长度，我们一定要在相关网站预先确认好。

3 → 按照 CA 机构规定的各种流程进行审核，审核内容包括：查看各种信贷数据、拨通在第三方机构数据库中记载的电话号码直接确认，等等。审核通过之后对 CSR 进行哈希处理，用 CA 机构的秘钥加密后将其作为数字签名添加到服务器证书中，接下来 CA 机构会发行服务器证书并交给请求方。服务器证书也是随机的字符串。

4 → 将从 CA 机构获得的服务器证书安装到 SSL 服务器上，有些 CA 机构会要求将中间证书也一起安装进去。中间证书指的是中间 CA 机构发行的一种证书。CA 机构为阶层性构造，管理着各色各样的证书，位于最顶层的是路由 CA 机构。中间 CA 机构是被 CA 机构（路由 CA 机构）认可的下属 CA 机构。

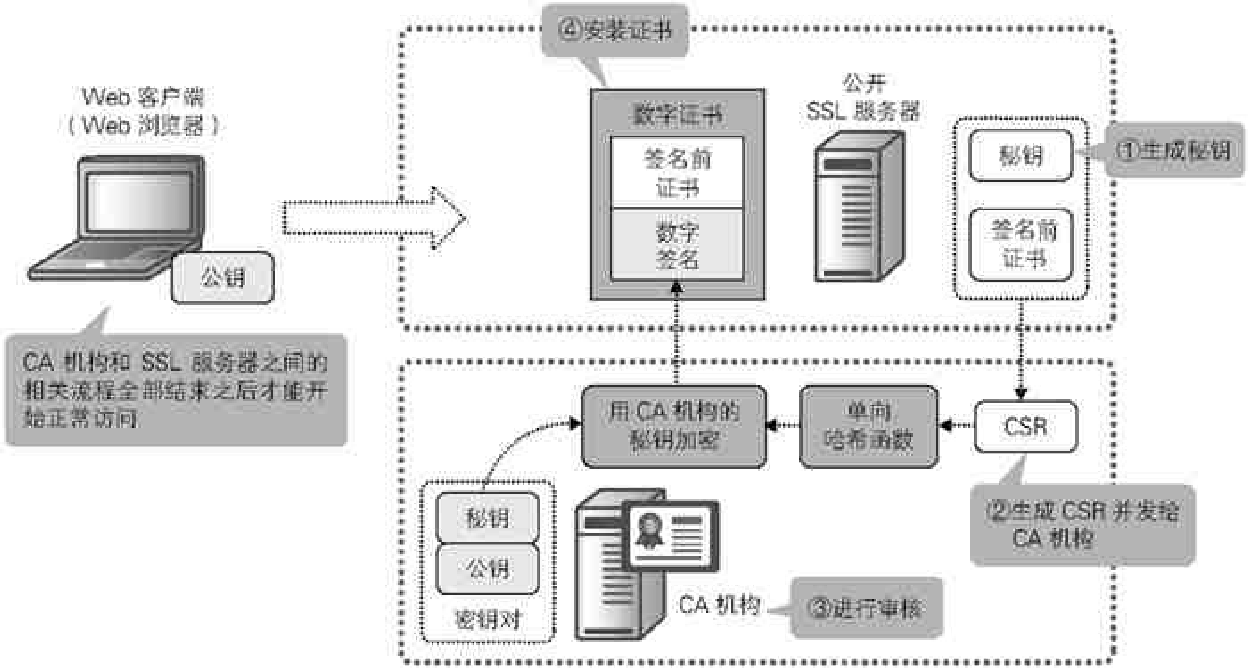


图 **3.2.28**　证书安装完毕之后，准备工作才算结束

大量处理结束之后，加密才得以进行

证书安装结束之后，接下来就要受理来自客户端的连接了。SSL 并不是马上给消息加密然后就发送出去的，在消息加密之前还有一个 SSL 握手阶段，用来决定给哪个信息加密。这里所说的握手和 TCP 的三次握手（依次为 SYN、SYN/ACK 和 ACK）是完全不同的概念。SSL 在 TCP 的三次握手结束之后再进行 SSL 握手处理，然后根据在该处理中决定的信息给消息加密。SSL 握手由四个步骤构成，分别是出示算法、证明通信对象的身份、交换共享密钥以及最终确认。

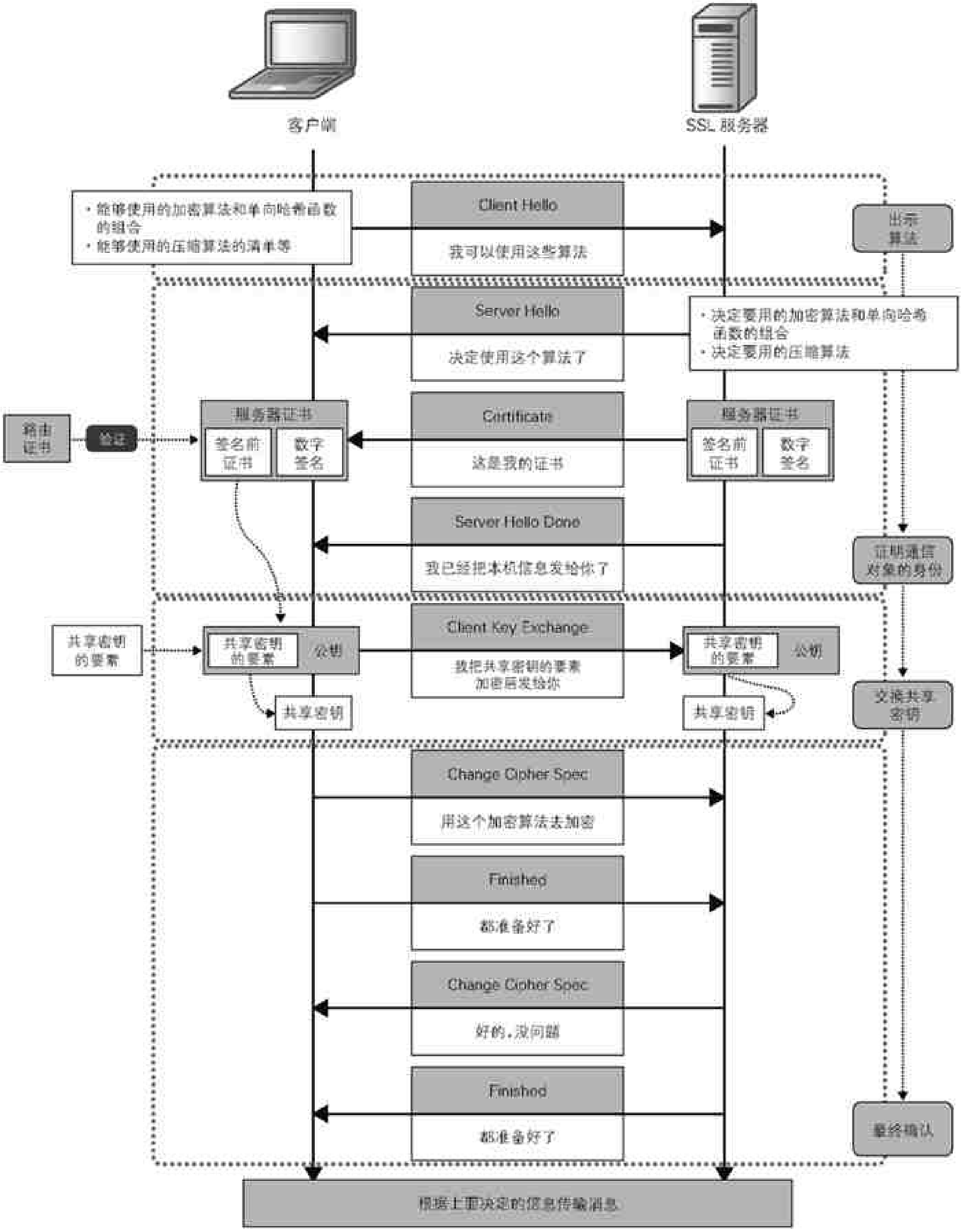


图 **3.2.29**　在 **SSL** 握手中交换共享密钥

1 →　出示支持的算法

在这个步骤中由客户端出示自己能够使用的加密方式和单向哈希函数。用于加密和哈希处理的技术（算法）非常之多，所以必须通过 Client Hello 告知对方本机能够使用哪些加密算法和单向哈希函数。

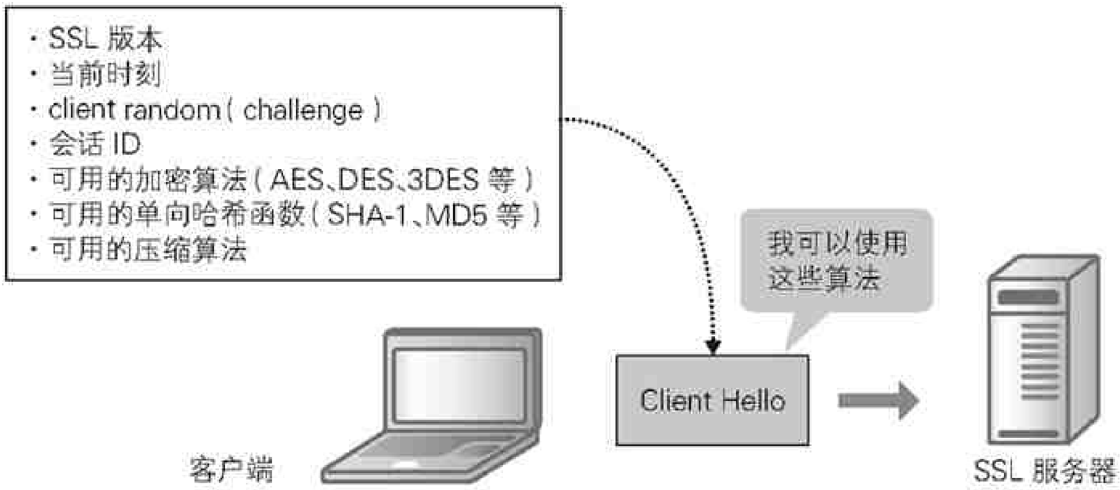


图 **3.2.30**　向服务器出示能够在本机中使用的参数

此外，在这个步骤中还要发送其他一些必须和服务器保持一致的参数，如 SSL 版本、会话 ID 和用于生成共享密钥的 client random 等。

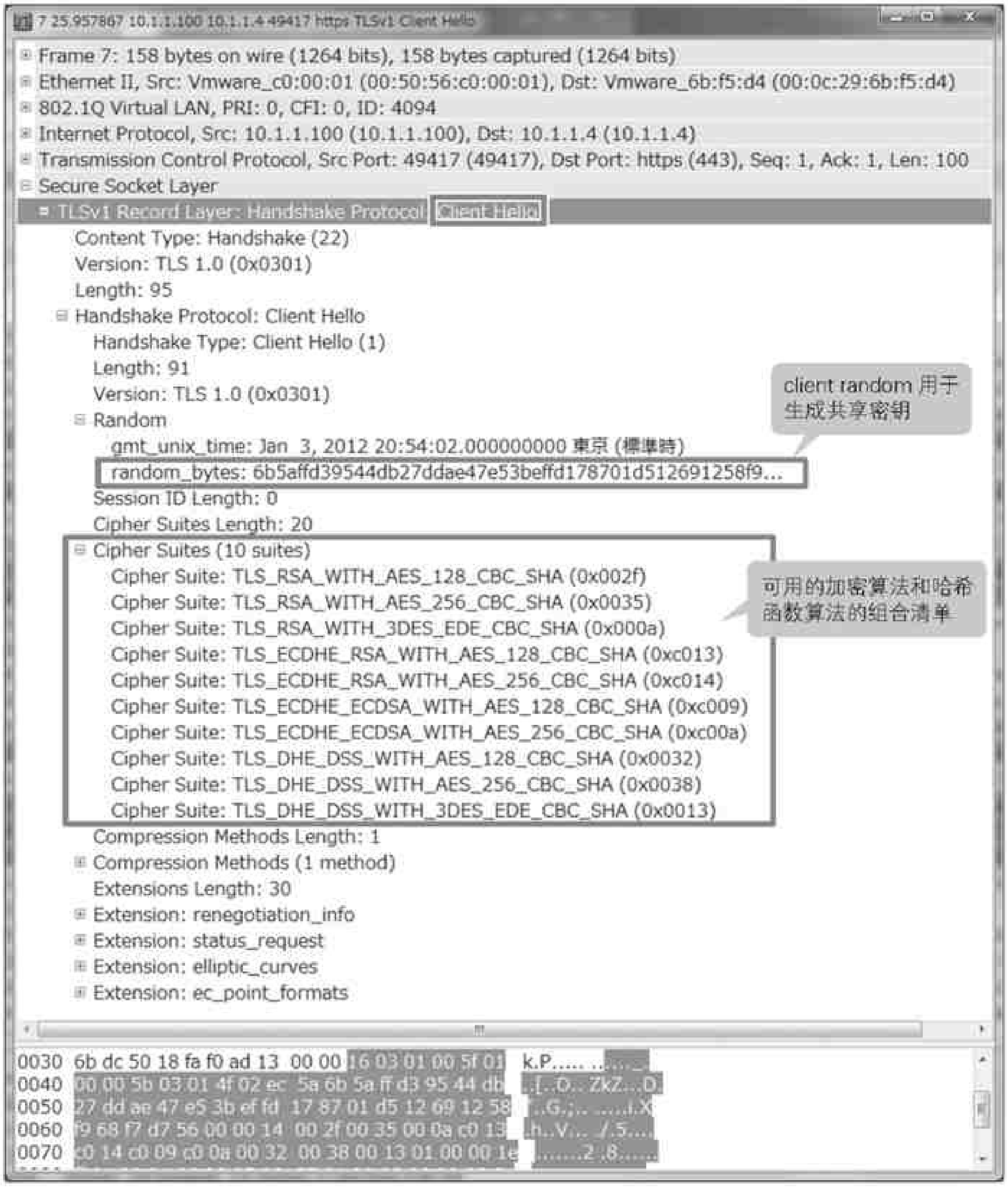


图 **3.2.31**　通过 **Client Hello** 向服务器出示支持的算法清单

2 →　证明通信对象的身份

在这个步骤中通过确认服务器证书来查明客户端是否在和真正的服务器通信，这个步骤由三个进程构成。

首先，服务器会在收到的清单中选出能够使用的算法，然后返回一个

Server Hello。除了选出的算法外，消息中还包括 SSL 版本、会话 ID、用于生成共享密钥的 server random 等必须和客户端保持一致的参数。接下来用 Certificate 发送本机的服务器证书，通知对方自己的身份。最后发出 Server Hello Done 的消息，告知对方所有信息都已发送完毕。客户端收到服务器证书后对其进行验证（用路由证书解密，然后比较双方的 MD 值），考察服务器的真伪。

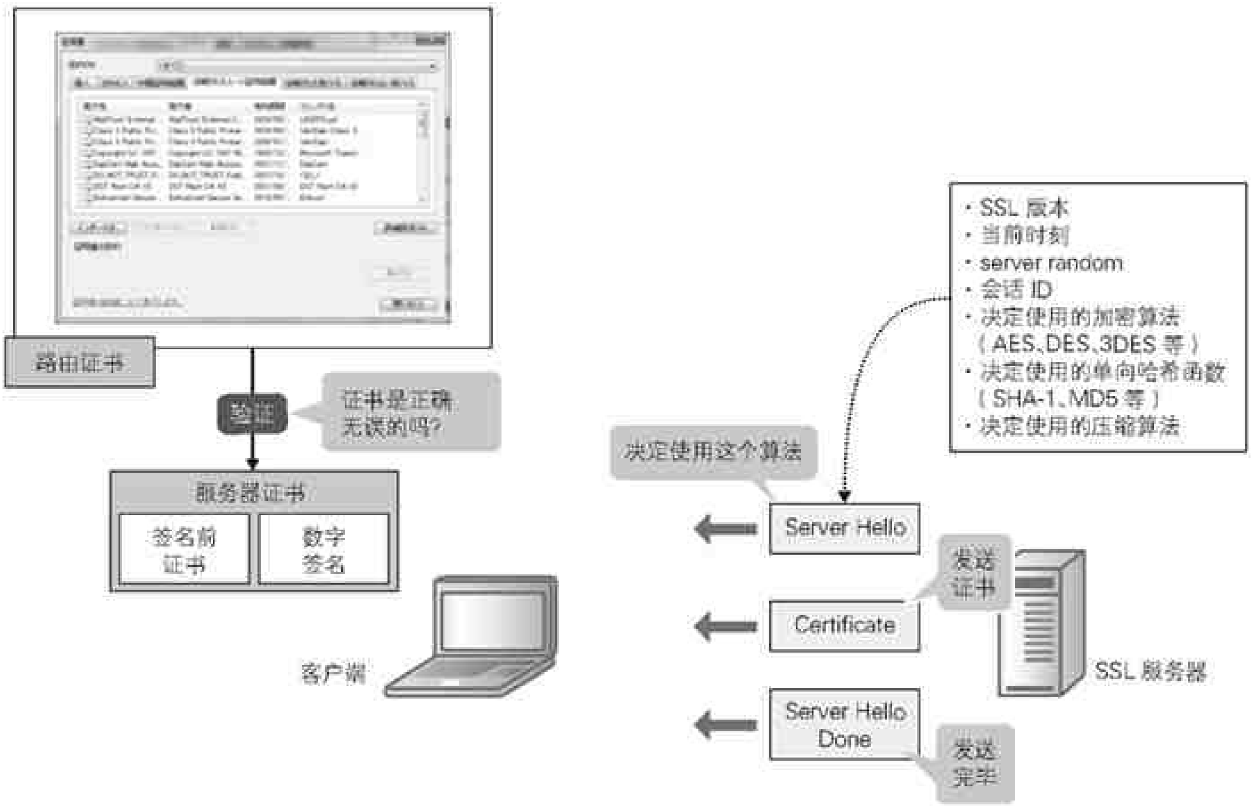


图 **3.2.32**　通过证书考察对方的身份

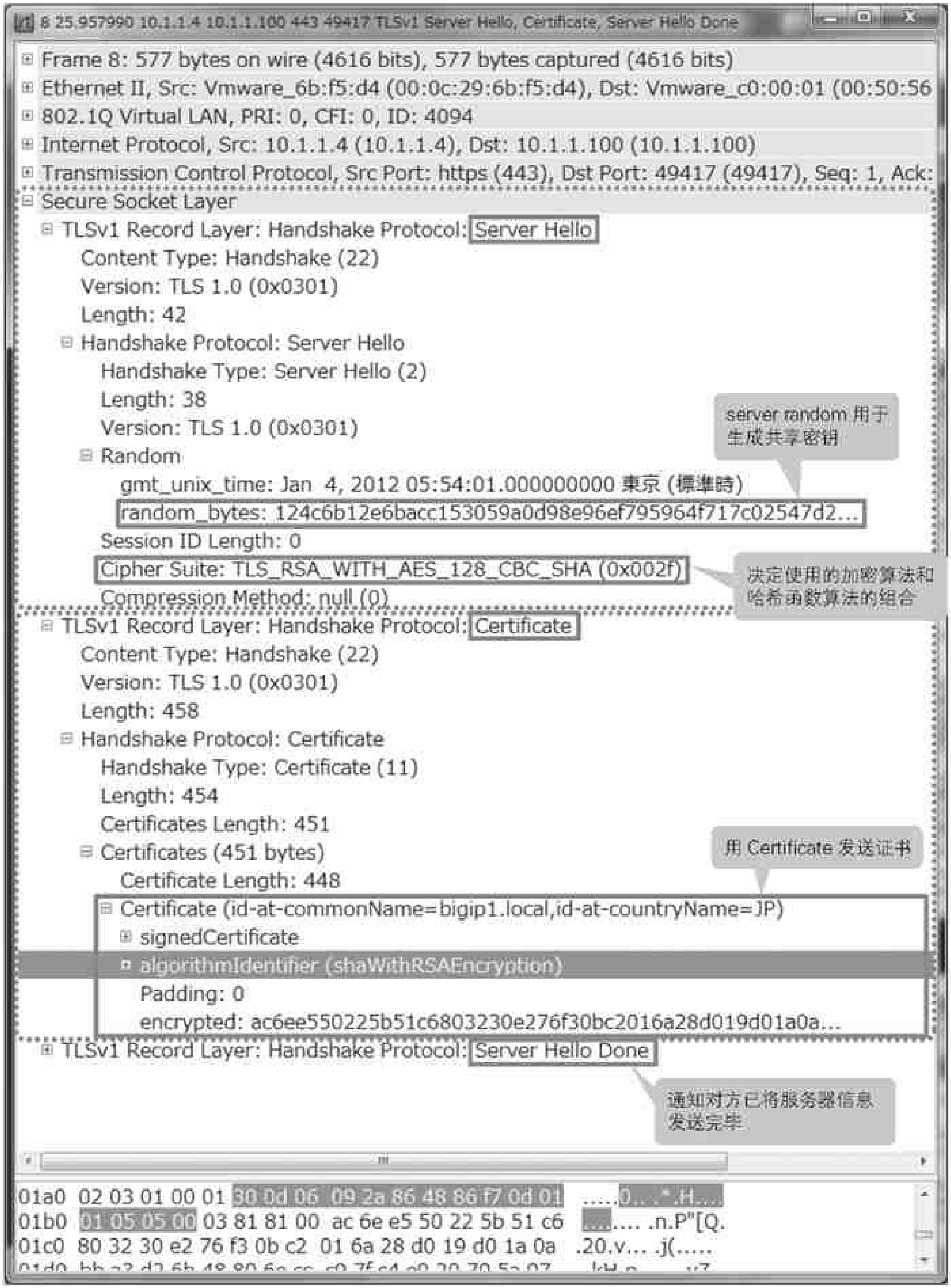


图 **3.2.33**　通过三个进程宣称自己的身份

3 →　交换共享密钥

在这个步骤中，双方交换用于加密消息的共享密钥。客户端确认服务器正确无误之后，会生成一个叫作预主秘钥的共享密钥要素，并将其发送给服务器。这个要素并非共享密钥，而只是组成共享密钥的素材，实际上使用的共享密钥是由预主密钥、通过 Client Hello 获得的 client random 以及通过 Server Hello 获得的 server random 混合而成的。client random 和 server random 由于在一开始就彼此交互，有着双方共同的部分，所以将密钥素材发送过去之后，就能够生成同样的共享密钥。

客户端在 Client Key Exchange 中用公钥给预主密钥加密，然后将其发送给服务器。服务器收到后再用秘钥为其解密，取出预主密钥之后将其与 client random 和 server random 合成，最终得到共享密钥。

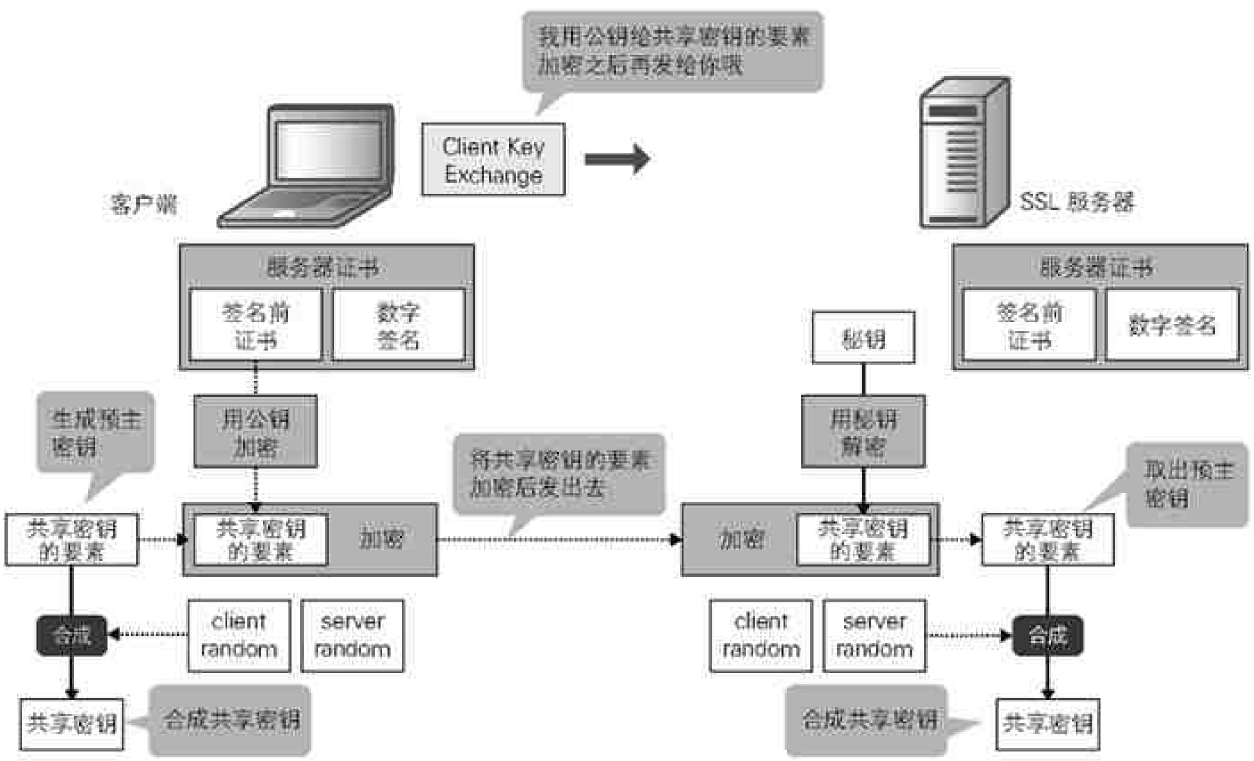


图 **3.2.34**　客户端将共享密钥的要素加密之后发给服务器

4 →　最终确认

这个步骤是最后的确认作业。双方交换 Change Cipher Spec 并宣称使用哪一种加密算法给消息加密。这一步的交互结束之后，才终于给传输加密并传输加密消息的阶段。

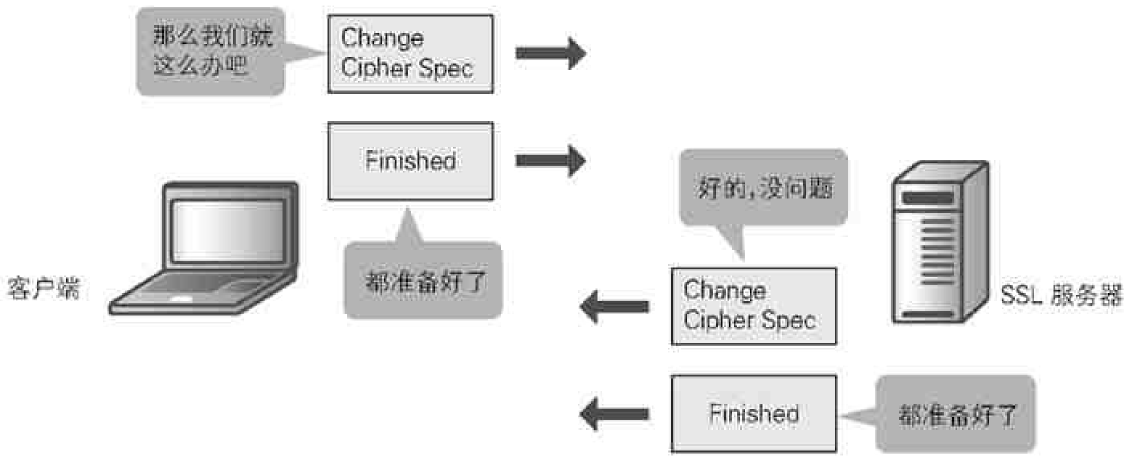


图 **3.2.35**　最后彼此进行确认

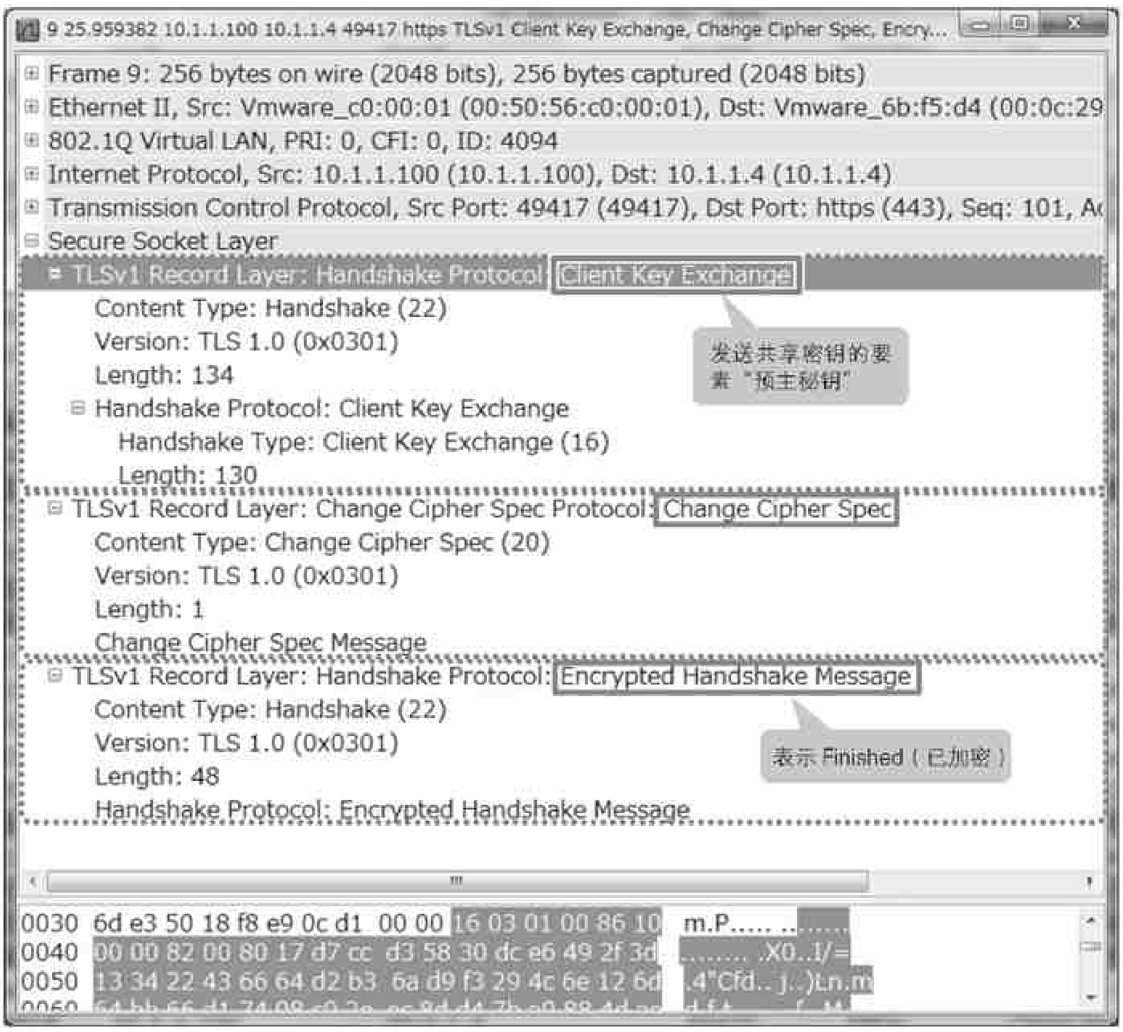


图 **3.2.36**　通过**Client Key Exchange**发送共享密钥的要素

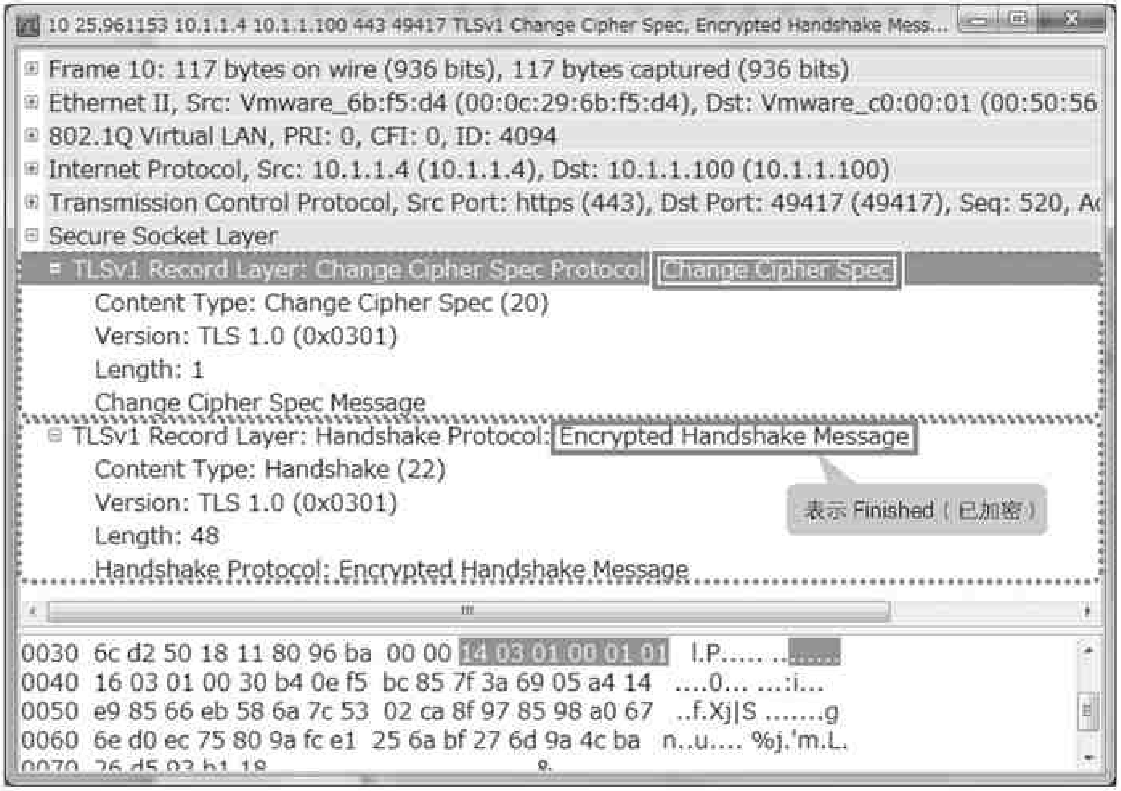


图 **3.2.37**　大量处理结束之后，加密才得以进行

3.2.2.6　用客户端证书对客户端进行认证

SSL 通信中包括两个认证机制，一个是服务器认证，另一个是客户端认证。服务器的认证要用到服务器证书，客户端的认证要用到客户端证书。

现在我们来着重看一下客户端认证。一般来说，客户端有两种认证方法，分别是密码认证和客户端认证。密码认证在 SNS 网站和在线购物网站中也会用到，大家应该都很熟悉。用户只要输入用户 ID 和密码，就会出现相应的网页。输入用户 ID 和密码的认证方法非常容易理解，而且无论身在何处，只要能够上网就可使用，若论方便，无出其右者。然而与此同时，这种方法也十分容易受到恶意攻击，一旦用户 ID 和密码泄露，谁都可以利用它们去冒充真正的用户。

客户端认证就是为了解决这个易受攻击的问题而出现的。该方法根据安装到客户端的客户端证书来识别对方是否为真正的用户，结论是肯定的才给予认证，也就是用证书取代了用户 ID 和密码。在 SSL 连接处理

中，SSL 服务器会要求客户端提供客户端证书。客户端返回代表本机身份的客户端证书之后，服务器会根据该证书的内容对客户端进行认证，通过之后才允许连接。双方互相发送己方的证书，彼此进行身份认证，安全性就能获得更高的保障。

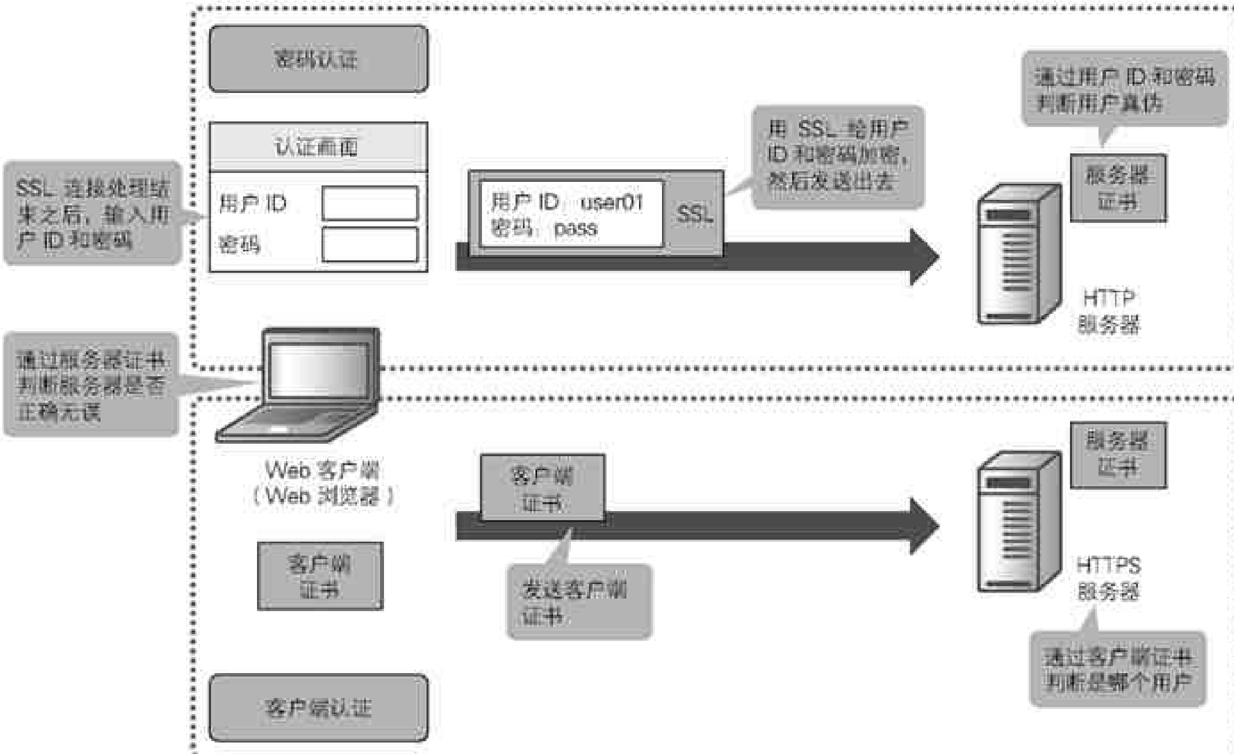
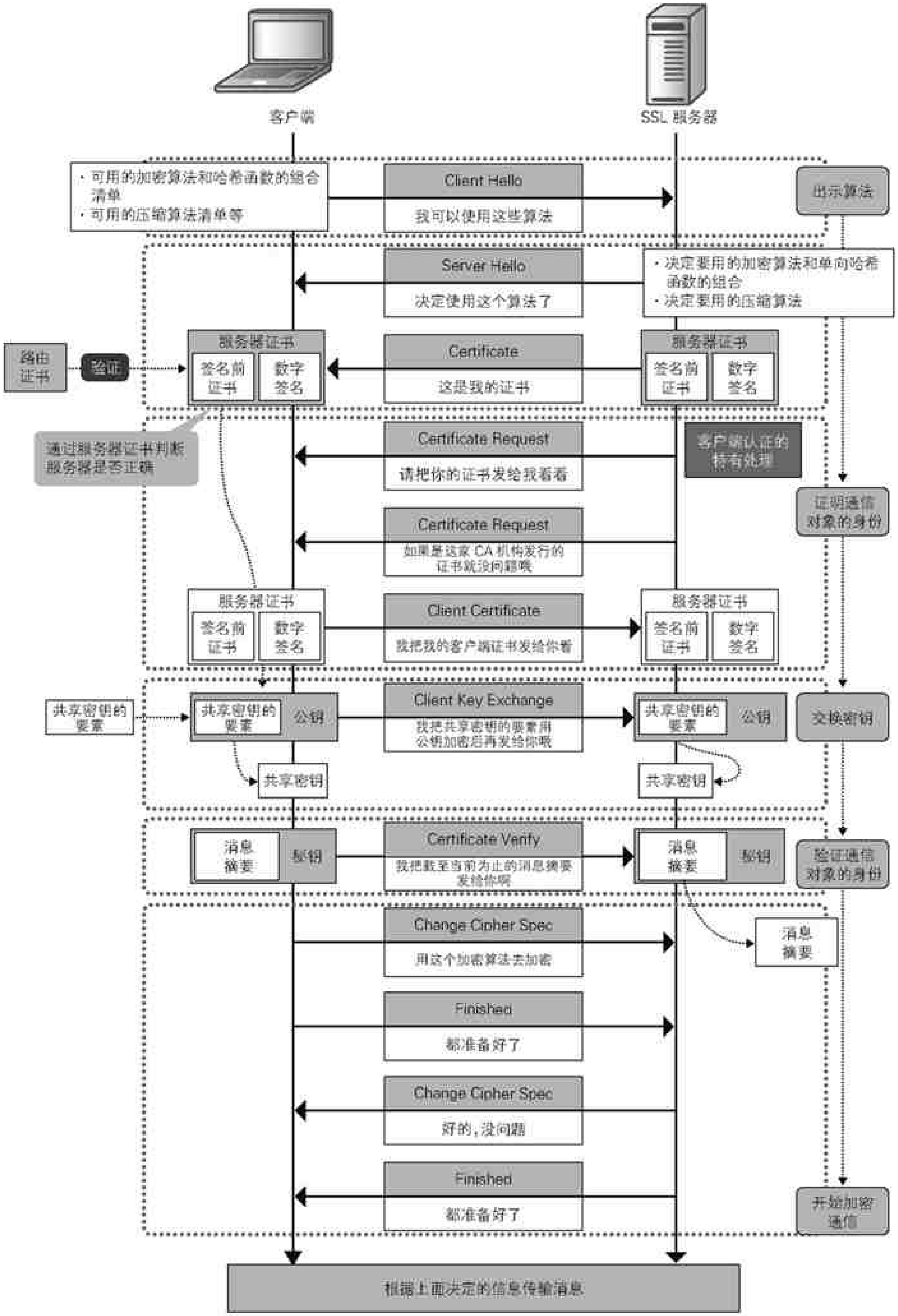


图 **3.2.38**　两种客户端认证机制

客户端认证的连接处理包括服务器认证及其他相关处理

除了前面讲述的服务器认证的握手处理之外，客户端认证的 SSL 握手处理还包括请求客户端证书和对客户端进行认证这两个进程。



客户端
SSL 服务器
· 可用的加密算法和哈希函数的组合清单
· 可用的压缩算法清单等
Client Hello
我可以使用这些算法
出示算法
Server Hello
决定使用这个算法了
· 决定要用的加密算法和单向哈希函数的组合
· 决定要用的压缩算法
路由证书
验证
服务器证书
签名前证书
数字签名
Certificate
这是我的证书
通过服务器证书判断服务器是否正确
Certificate Request
请把你的证书发给我看看
客户端认证的特有处理
证明通信对象的身份
Certificate Request
如果是这家 CA 机构发行的证书就没问题哦
Client Certificate
我把我的客户端证书发给你看
共享密钥的要素
公钥
Client Key Exchange
我把共享密钥的要素用公钥加密后再发给你哦
交换密钥
共享密钥
消息摘要
秘钥
Certificate Verify
我把截至当前为止的消息摘要发给你啊
验证通信对象的身份
Change Cipher Spec
用这个加密算法去加密
Finished
都准备好了
Change Cipher Spec
好的，没问题
Finished
都准备好了
开始加密通信
根据上面决定的信息传输消息

图 **3.2.39**　除握手处理之外还要执行对客户端进行认证的交互处理

1 →　请求客户端证书

首先要说明的是，从一开始到发送服务器证书的 Certificate 步骤为止全都和服务器认证的步骤一样。但接下来，服务器将本机的服务器证书发给客户端之后，会通过 Certificate Request 要求客户端也将客户端证书发送过来，而且会出示可以信任的 CA 机构清单。

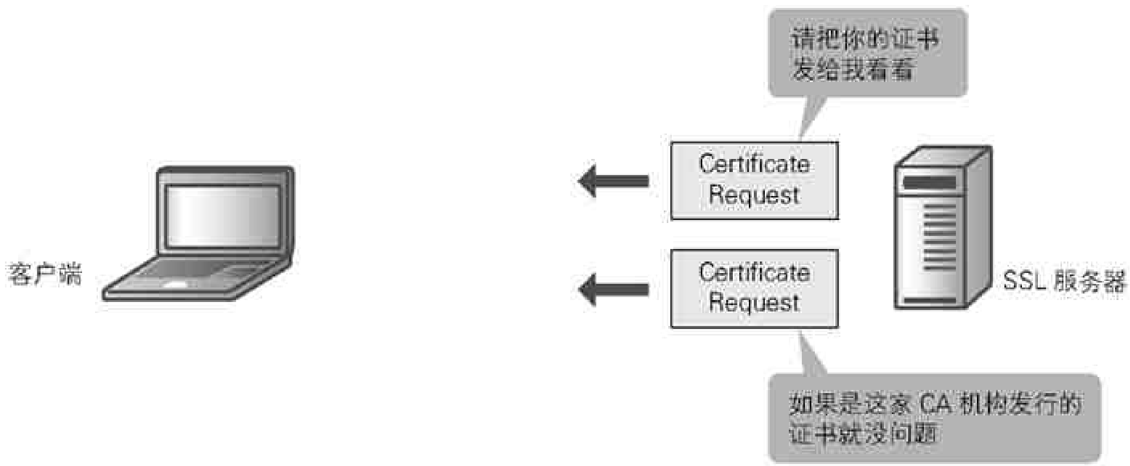


图 **3.2.40**　要求发送客户端证书

2 →　发送客户端证书

作为回应，客户端会通过 Client Certificate 将已安装到本机上的客户端证书发送给服务器。如果没有符合服务器要求的客户端证书，则返回 no_certificate，然后服务器就会断开连接；如果有多个符合要求的客户端证书，则在浏览器上选择其中一个发送。



图 **3.2.41**　发送客户端证书

3 →　验证通信对象的身份

接下来，客户端在 Client Key Exchange 中将预主密钥发给服务器，这和通常的处理是一样的。然后，再通过 Certificate Verify 算出从 Client Hello 到 Client Key Exchange 为止交互的消息摘要，用秘钥加密之后发给服务器。服务器收到 Certificate Verify 之后，用通过 Client Certificate 收到的客户端证书中所含的公钥为其解密，然后将收到的消息摘要和自己计算出来的消息摘要进行对比，查看数据是否被篡改过，随后的处理都和通常处理一模一样。最后，根据上面决定的信息给消息加密然后发送出去。

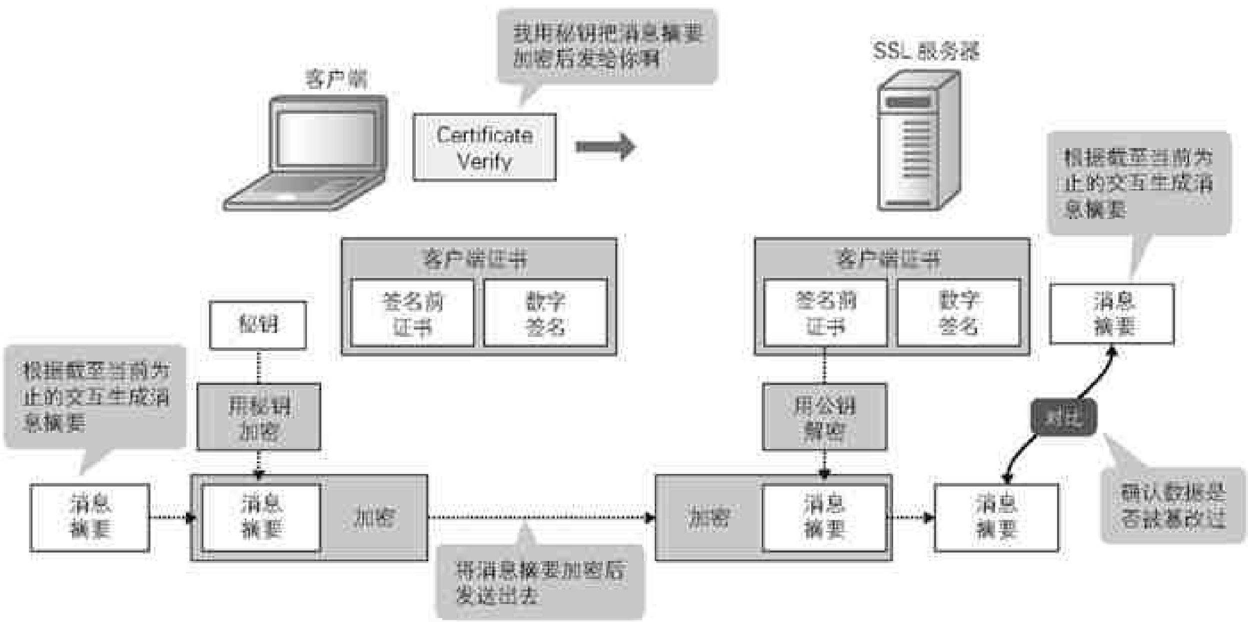


图 **3.2.42**　将截至当前为止的两个消息摘要进行对比

3.2.3　用 **FTP** 传输文件

正如其名，FTP（File Transfer Protocol，文件传输协议）是一种用于传输文件的协议。尽管它本身并不具备加密功能而无法保证数据安全，但是至今仍在很多地方广为使用。



图 **3.2.43　FTP** 本身并不具备加密功能

完整的 FTP 传输是将两种连接组合起来使用的，一种叫作控制连接，另一种叫作数据连接。控制连接用于控制应用程序，人们用它发送命令或返回结果；数据连接则用于实现真正的数据传输。每一条用控制连接发出的命令都会生成相应的数据连接，在数据连接上收发数据。

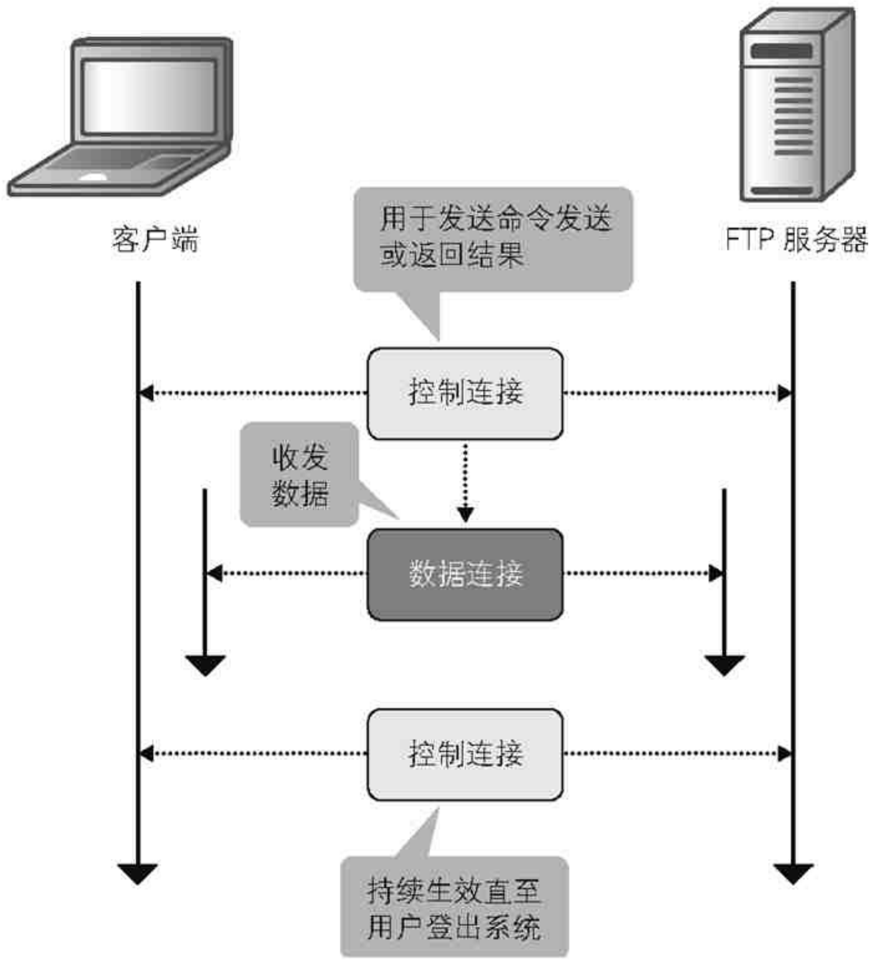


图 **3.2.44　FTP** 将两种连接组合起来使用

说到 FTP，必须要提一个叫作传输模式的重要概念。FTP 中有两种传输模式，一个是主动模式，另一个是被动模式，二者生成数据连接的方式有些不太一样。下面，本书将着重结合所用端口号和连接请求（SYN）方向这两个要素来介绍数据连接。

3.2.3.1　主动模式使用特定的端口

FTP 的主动模式是指在控制连接中使用 TCP/21、在数据连接中使用 TCP/20 的模式。大多数 FTP 客户端软件的默认设置都是主动模式，在命令提示符中使用的、Windows 标准的 FTP 客户端功能中甚至只有这一种模式。

主动模式是由服务器方面发出数据连接的连接请求（SYN）的，是一种特殊的传输模式。几乎所有的主从式（客户端—服务器）架构协议都是

由客户端发出连接请求，然后由服务器方面给出回复。然而在主动模式的数据连接中，这个执行方向恰恰相反，先是由服务器发出连接请求，然后由客户端给出回复。那么，我们就假设客户端需要从 FTP 服务器获取某个文件（RETR）[9]，来看看实际的连接步骤是怎样的。

> [9] RETR 相当于 HTTP 中的 GET 方法，下载文件时要用到它。

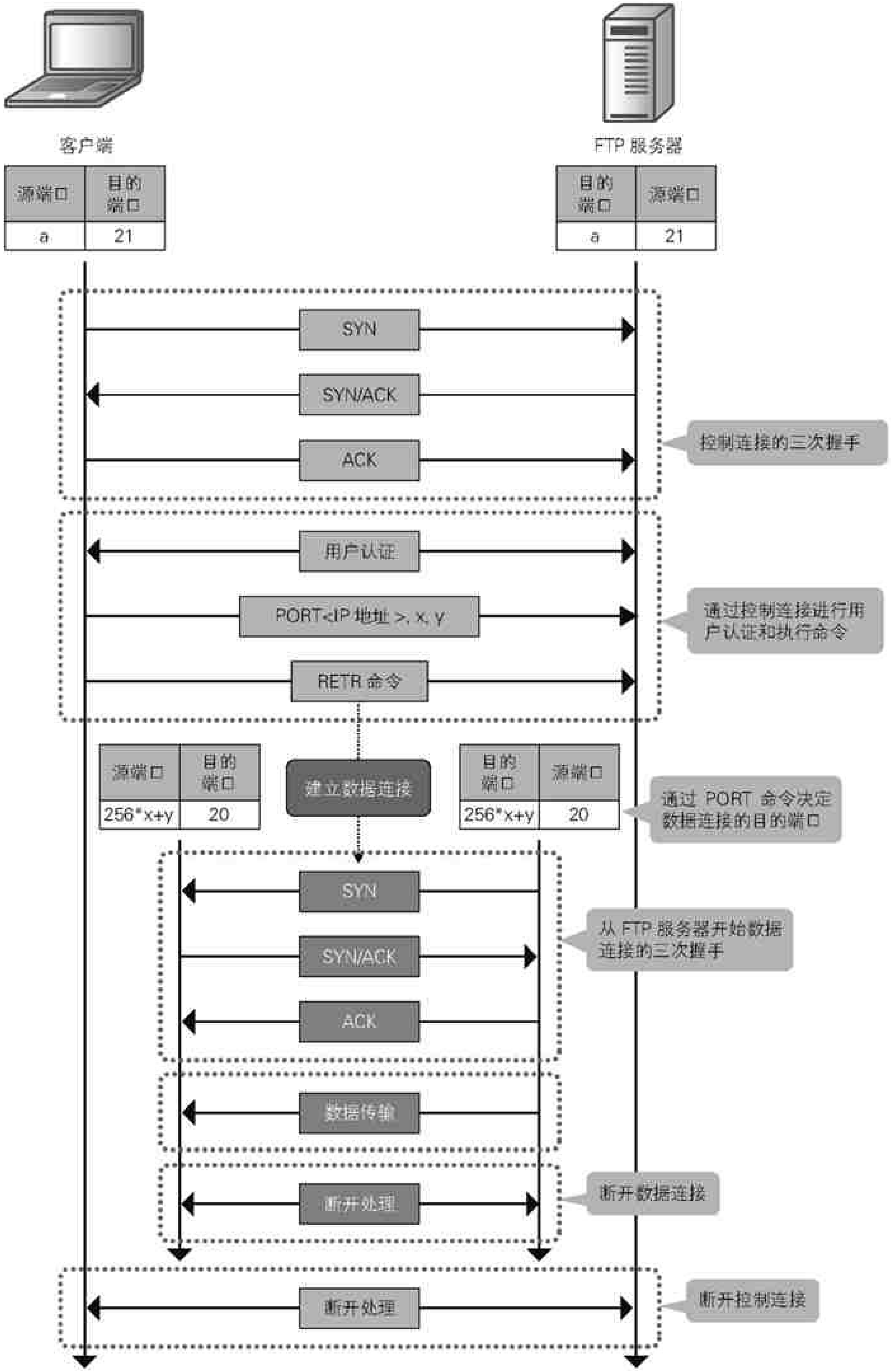


图 **3.2.45**　主动模式从服务器一方开始建立数据连接

1 →　客户端向服务器提出用 TCP/21 连接的请求，三次握手结束。到这里还只是建立了控制连接。

2 →　在控制连接上交换用户名和密码之后，客户端通过 PORT 命令发送用于数据连接的端口号要素值，该值以“PORT<IP 地址 >,x,y”的形式发送。这里重要的并不是 IP 地址而是 x 和 y 的值，用公式“256*x＋y”算出来的值就是数据连接的目的端口号，将会在步骤 **4 →** 中使用。

为了帮助大家理解，这里举个例子解释一下。假设客户端已通过 PORT 命令发送了 x＝150、y＝218 这两个值（如下图），那么，即将建立的数据连接的目的端口号就是 150×256＋218＝38618。



图 **3.2.46**　通过 **PORT** 命令发送数据连接的端口号要素值

※ 图中的 Active port 值是由 Wireshark 自动计算出来的，这个值不会发送给服务器。

3 →　客户端通过控制连接向服务器发送 RETR（GET）命令。

4 →　服务器收到 RETR 命令之后，在源端口号中写入 TCP/20，在目的端口号中写入通过 PORT 命令算出的值（256*x＋y）并请求连接。这里的连接请求是关键所在，它是由服务器一方提出来的。TCP/20 的三次握手结束之后，就能将应用数据发送出去了。

5 →　应用数据一经发送就执行 TCP 断开处理，关闭数据连接，但控制连接仍继续生效。

6 →　最后在用户登出系统的同时执行 TCP/21 断开处理，关闭控制连接。至此，所有的处理才宣告结束。

3.2.3.2 被动模式改变使用的端口

FTP 的被动模式是指在控制连接中使用 TCP/21、在数据连接中使用不特定端口的一种模式。最近人们出于在数据安全方面的一些考虑，使用这种模式的例子逐渐多了起来。

被动模式仅当客户端通知服务器它处于被动模式（PASV 命令）时才会启用。数据连接的连接请求（SYN）是从客户端发出的，从这一点来说，几乎所有的主从式（客户端—服务器）架构协议都是如此。被动模式的关键在于它的端口号，主动模式一定会用 TCP/20，被动模式则会选择非特定的端口来使用。那么，我们就假设客户端需要从 FTP 服务器获取某个文件（RETR 命令），来看看实际的连接步骤是怎样的。

1 → 客户端向服务器提出用 TCP/21 连接的请求，三次握手结束。到这里还只是建立了控制连接。

2 → 在控制连接上交换用户名和密码之后，客户端通过 PASV 命令提出使用被动模式的请求。

针对该请求，服务器会返回一个 Entering Passive mode 的提示，同时发送用于数据连接的端口号要素值，该值以“Entering passive mode<IP 地址 >,x,y”的形式发送。这里重要的并不是 IP 地址而是 x 和 y 的值，用公式“256*x＋y”算出来的值就是数据连接的目的端口号，将会在步骤 **3 →** 中使用。

为了帮助大家理解，这里举个例子解释一下。假设客户端已通过 Entering passive mode 命令发送了 x＝212、y＝174 这两个值（如下图），那么，即将建立的数据连接的目的端口号就是 212×256＋174＝54446。

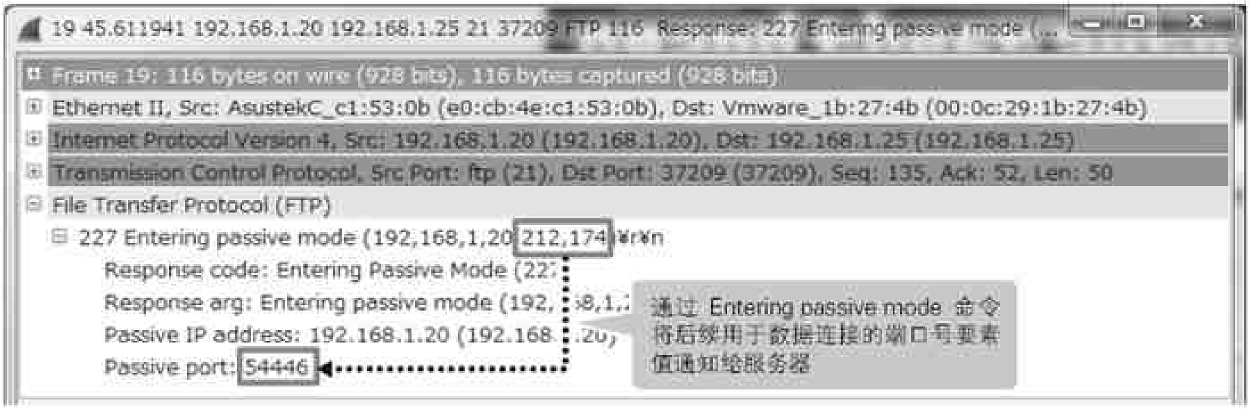


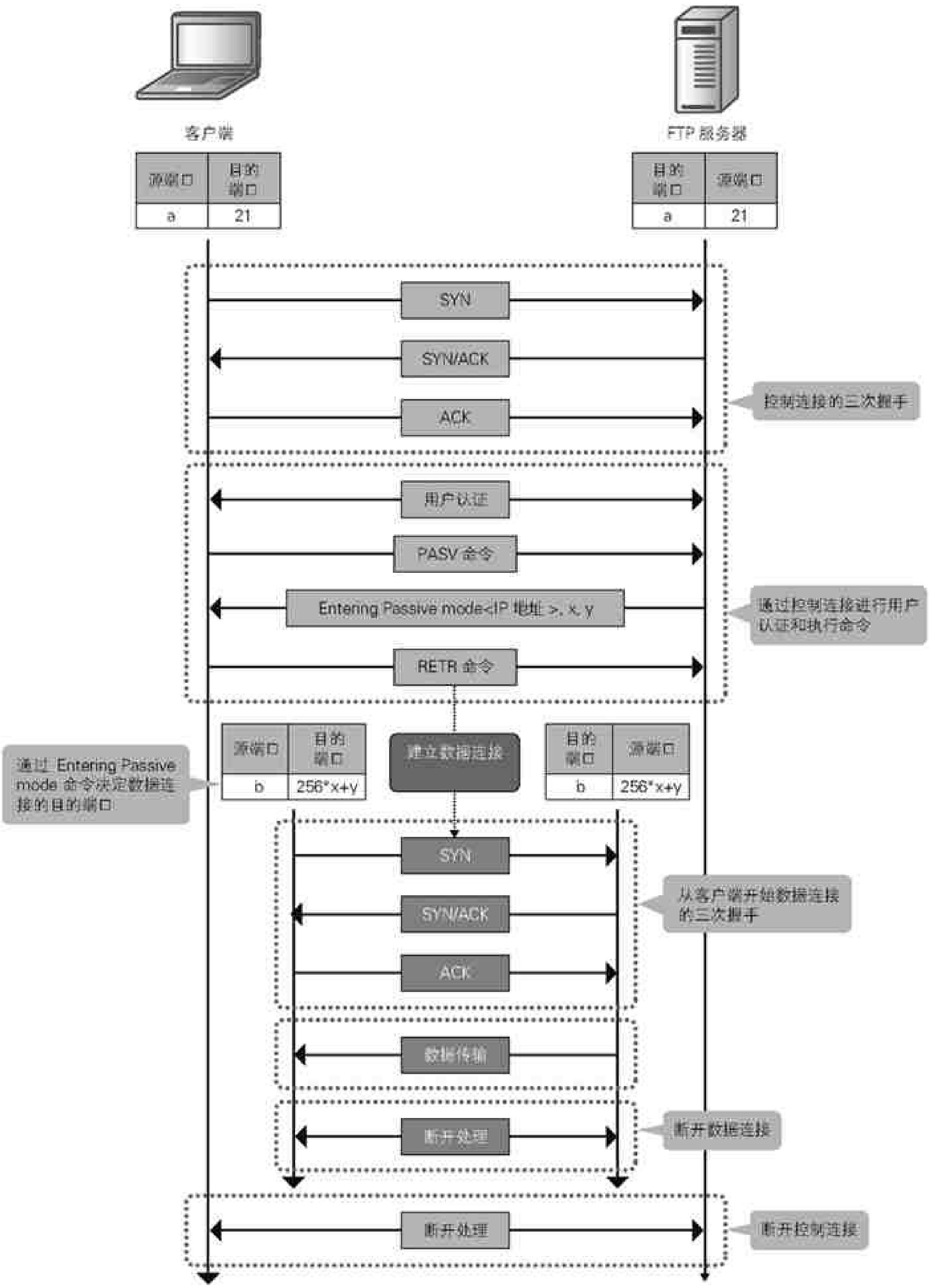
图 **3.2.47**　通过 **Entering passive mode** 命令发送数据连接的端口号要素值

* 图中的 Passive port 值是由 Wireshark 自动计算出来的，这个值不会发送给服务器。

3 →　客户端使用控制连接向服务器发送 RETR 命令。然后在目的端口号中写入前面通过 Entering passive mode 命令计算出来的端口号，提出连接请求并生成数据连接。顺便提一句，这个时候的源端口号是随机的。三次握手结束之后数据连接得以建立，这样就能在数据连接上发送应用数据了。

4 →　应用数据一经发送就执行 TCP 断开处理，关闭数据连接，但控制连接仍继续生效。

5 →　最后在用户登出系统的同时执行 TCP/21 断开处理，关闭控制连接。至此，所有的处理才宣告结束。



客户端
FTP 服务器
源端口
目的端口
a
21
目的端口
源端口
a
21
SYN
SYN/ACK
ACK
控制连接的三次握手
用户认证
PASV 命令
Entering Passive mode<IP 地址 >, x, y
通过控制连接进行用户认证和执行命令
RETR 命令
源端口
目的端口
b
256*x+y
建立数据连接
目的端口
源端口
b
256*x+y
通过 Entering Passive mode 命令决定数据连接的目的端口
SYN
SYN/ACK
ACK
从客户端开始数据连接的三次握手
数据传输
断开处理
断开数据连接
断开处理
断开控制连接

图 **3.2.48**　被动模式从客户端开始建立数据连接

3.2.3.3　FTP 就应该当作 FTP 去处理

FTP 是一种比较特殊的应用协议，要用到多个端口号的各种组合。因此，如果我们在防火墙或负载均衡器中只是把它单纯地当作 TCP 连接去处理（TCP/21、TCP/20），很可能会产生前后无法呼应的现象。FTP 就应该当作 FTP 去处理，而且一定要在应用程序的层面上处理才行。

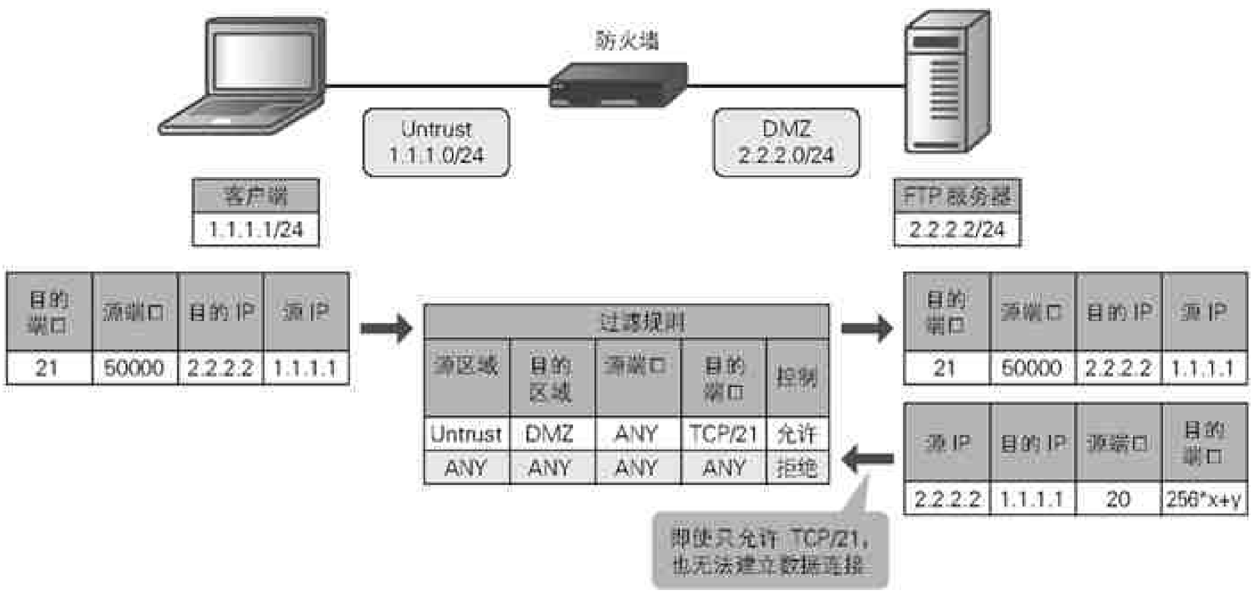


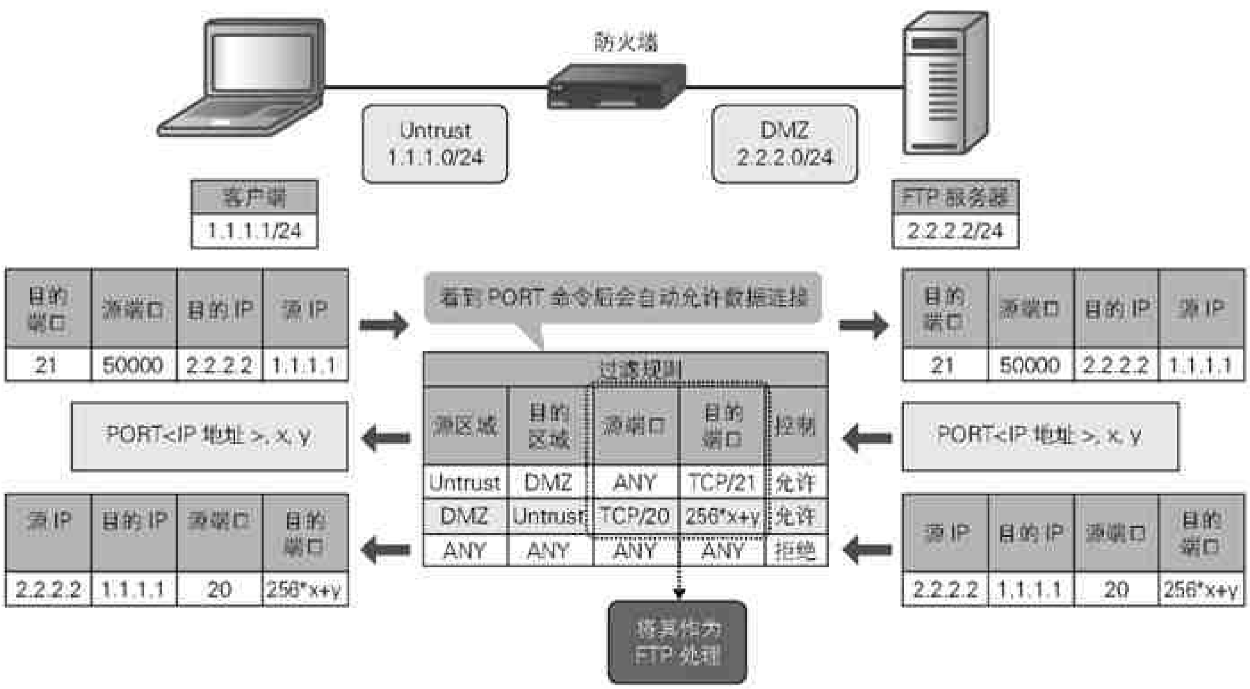
图 **3.2.49**　如果将 **FTP** 当作 **TCP** 去处理，很可能会产生前后无法呼应的现象

“FTP 就应该当作 FTP 去处理，而且一定要在应用程序的层面上处理”这句话听起来似乎有些高深，但其实工作原理并没有那么复杂。这里所说的“应用程序的层面”可以理解为 FTP 命令。防火墙和负载均衡器监控着通过 PORT、PASV 和 Entering passive mode 命令交互的端口号信息，同时也一直在动态地等待着后续需要处理的数据连接。只要收到数据连接的 SYN 包，它们就会马上用等待端口去进行处理。

“将 FTP 当作 FTP 处理”还涉及一个功能，那就是保持控制连接的功能。我们通过数据连接传输数据的时候，控制连接是用不上的。然而，如果传输的数据很大导致耗时较长，那么控制连接可能会因为连接空闲时间超过一定限度而断掉。为了避免出现这种情况，人们想办法做了一些处理，让控制连接的连接空闲时间在传输数据的过程中绝不会超过时

间限制。

谈到防火墙或负载均衡器中的设置，其实几乎所有的设备中都定义了用于识别 FTP 的信息（配置文件）。而且，有些设备只要收到来自 TCP/21 的连接就会将其当作 FTP 处理。如果我们让这些设置都生效，设备就会主动去查看命令内容并执行各种处理。



目的端口	源端口	目的 IP	源 IP
21	50000	2.2.2.2	1.1.1.1

源区域	目的区域	源端口	目的端口	控制
Untrust	DMZ	ANY	TCP/21	允许
DMZ	Untrust	TCP/20	256*x+y	允许
ANY	ANY	ANY	ANY	拒绝

源 IP	目的 IP	源端口	目的端口
2.2.2.2	1.1.1.1	20	256*x+y

图 **3.2.50**　对 **FTP** 施以它作为 **FTP** 本应承受的处理（图为防火墙的示例）

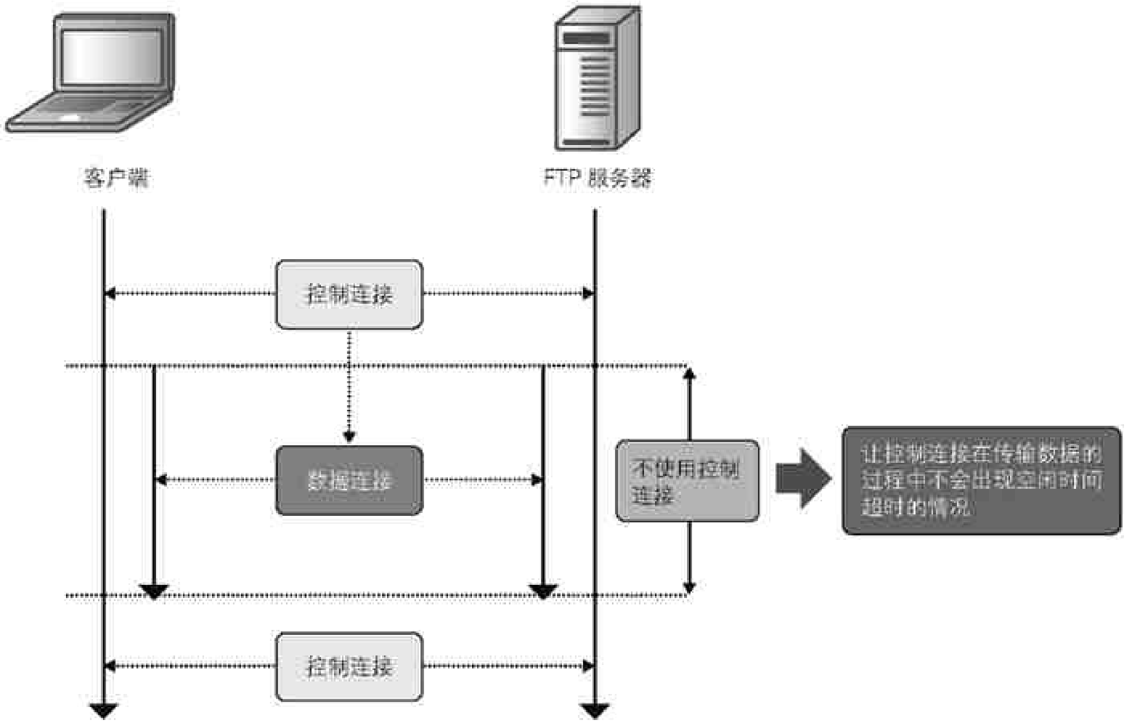


图 **3.2.51**　让控制连接的连接空闲时间不会超时

3.2.4　用 **DNS** 解析名称

DNS（Domain Name System，域名系统）是一种用于解析名称的协议。互联网上分布着众多 IP 地址，但是我们不可能为了浏览网站而去一一记住 IP 地址的那些长串数字。DNS 就是为了解决这个问题应运而生的，它为每个 IP 地址都取了域名，这样人们记忆起来就容易多了。关于 DNS 的构造以及应该如何设置服务器这些深层次的内容，笔者建议各位读者去查阅专业书籍，而本书仅从网络这个角度对 DNS 作一些简单的介绍。

DNS 会根据不同的用途选择使用 UDP 或 TCP。二者都属于普通常见的连接，下面我们来看看它们分别用于什么场合。

3.2.4.1　用 **UDP** 进行名称解析

名称解析一般会先于网络、邮件等应用通信进行，因此速度是否够快非常重要。UDP 的实时性很高，人们在追求速度时往往会选择它。其实，我们在浏览器上查看网站内容时，并不是通过该网站的 HTTP 直接进行访问的。另外，我们在发送邮件的时候，也不是直接把邮件发送给

邮件服务器的。这两种情况都必须先经过一个叫作名称解析的步骤。

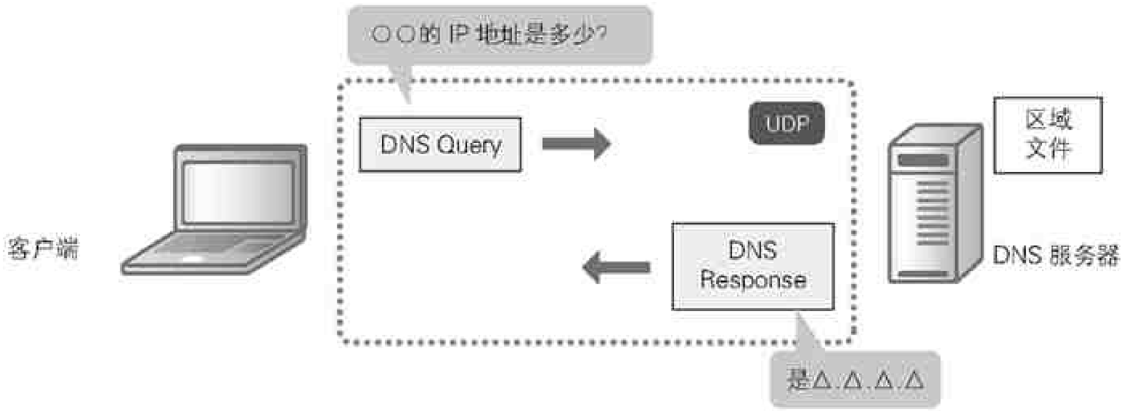


图 **3.2.52**　名称解析通过 **UDP** 追求实时性

1 →　客户端发通过 UDP 的 DNS 包向 DNS 服务器查询域名。

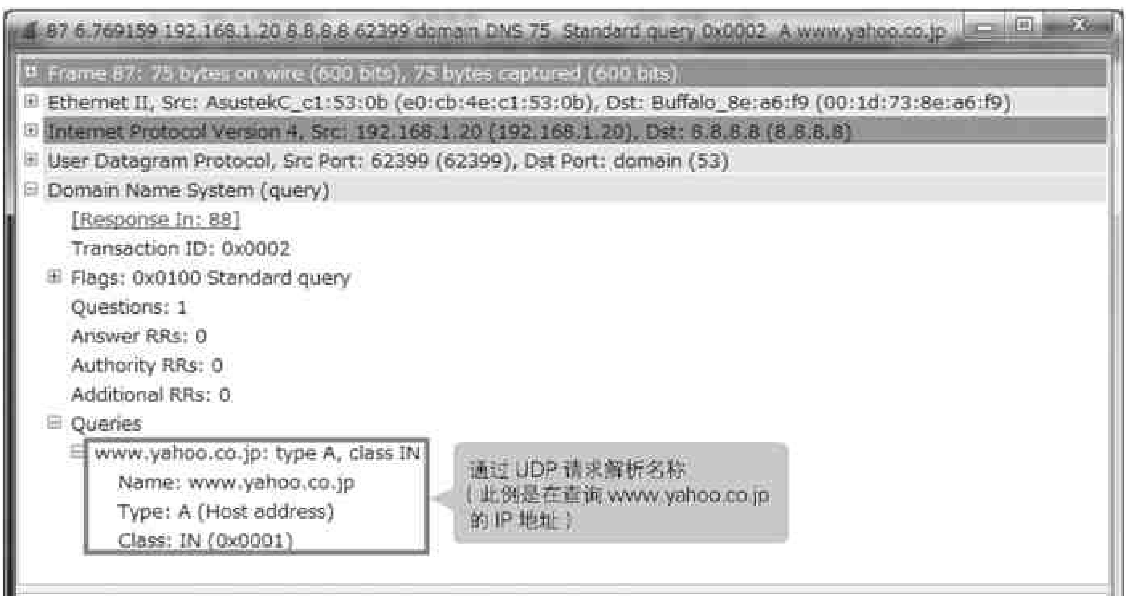


图 **3.2.53**　通过 **UDP** 请求解析名称

2 →　DNS 服务器中有区域文件，IP 地址和域名信息就以区域文件的形式保存在其中。收到查询后，DNS 服务器从该文件中找到相应的 IP 地址并将其返回给客户端。

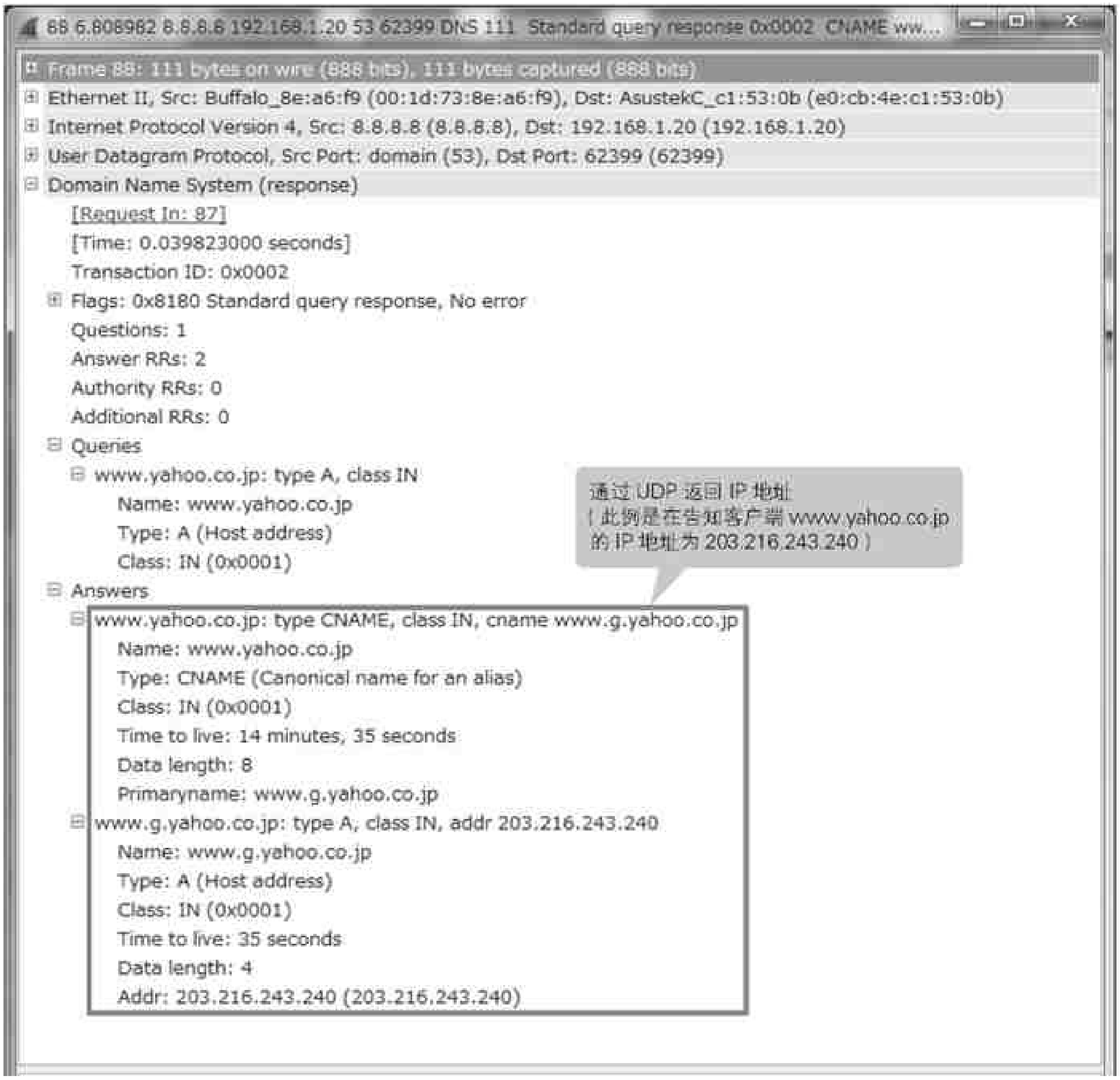


图 **3.2.54**　通过 **UDP** 返回 **IP** 地址

3 →　客户端通过 HTTP 访问该 IP 地址或者发送邮件。

3.2.4.2　用 **TCP** 进行区域传输

DNS 是很多应用程序的幕后英雄，是非常重要的应用协议，在这里发生问题的话就无法进入后续的应用通信阶段，所以人们一般会对 DNS 服务器进行冗余配置以保证稳定地提供名称解析服务。区域传输就是用于 DNS 服务器冗余配置的一项功能。DNS 服务器以区域文件的形式保存着 IP 地址和域名信息，区域传输就是用来同步此文件的功能。通过区域传输功能在主服务器和从属服务器之间同步区域文件，以此来保证冗余效果。主服务器如果宕机，就通过从属服务器的区域文件给出回

复。区域传输并不需要实时进行，对它来说可靠性才是最重要的，因此人们一般选择使用 TCP 来实现这个功能，具体步骤如下图所示。请注意，这里画的只是 DNS 服务器的事实标准——BIND 软件的工作机制。

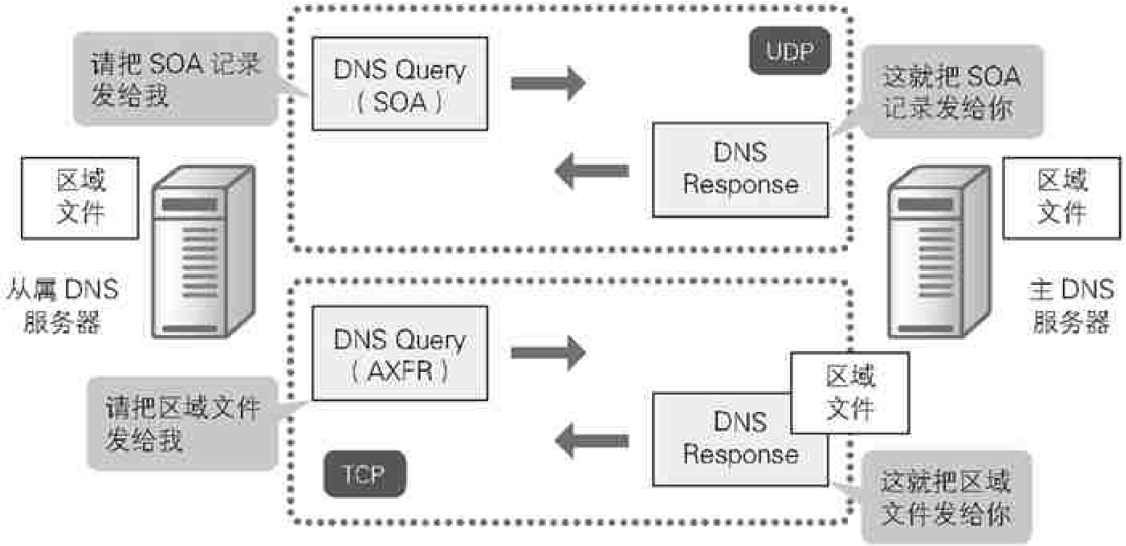


图 **3.2.55**　用 **TCP** 进行区域传输

1 →　当区域文件的有效期已过或是收到 notify 消息时，从属服务器会通过 UDP/53 请求主服务器发送 SOA 记录。SOA 记录中包含着 DNS 服务器的管理信息。

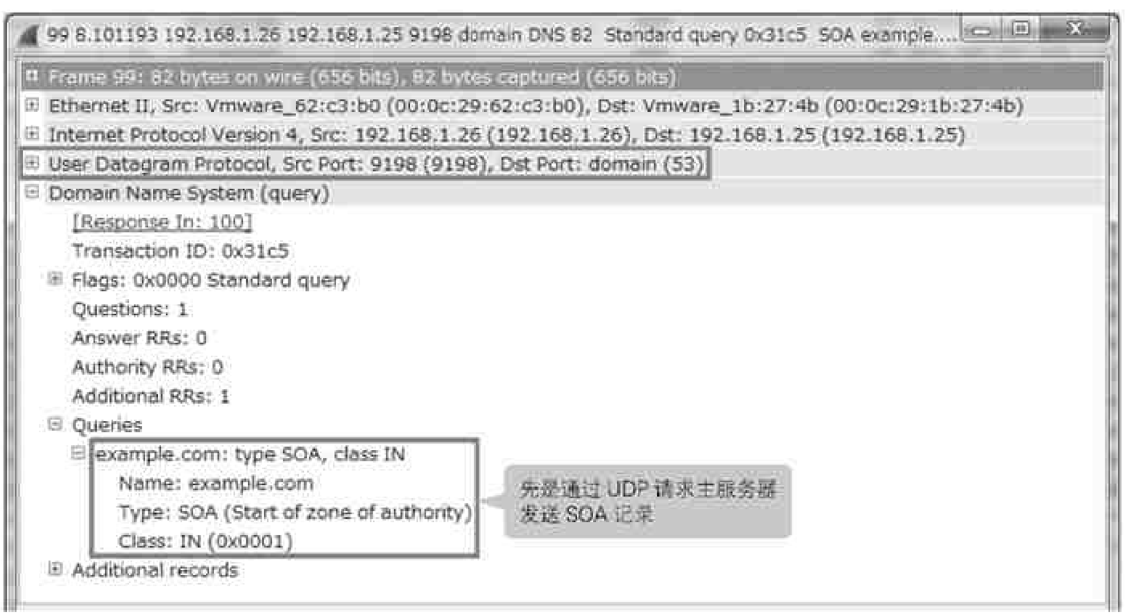


图 **3.2.56**　先是通过 **UDP** 请求发送 **SOA** 记录

2 →　主服务器通过 UDP/53 将区域文件中的 SOA 记录返回给从属服务器。

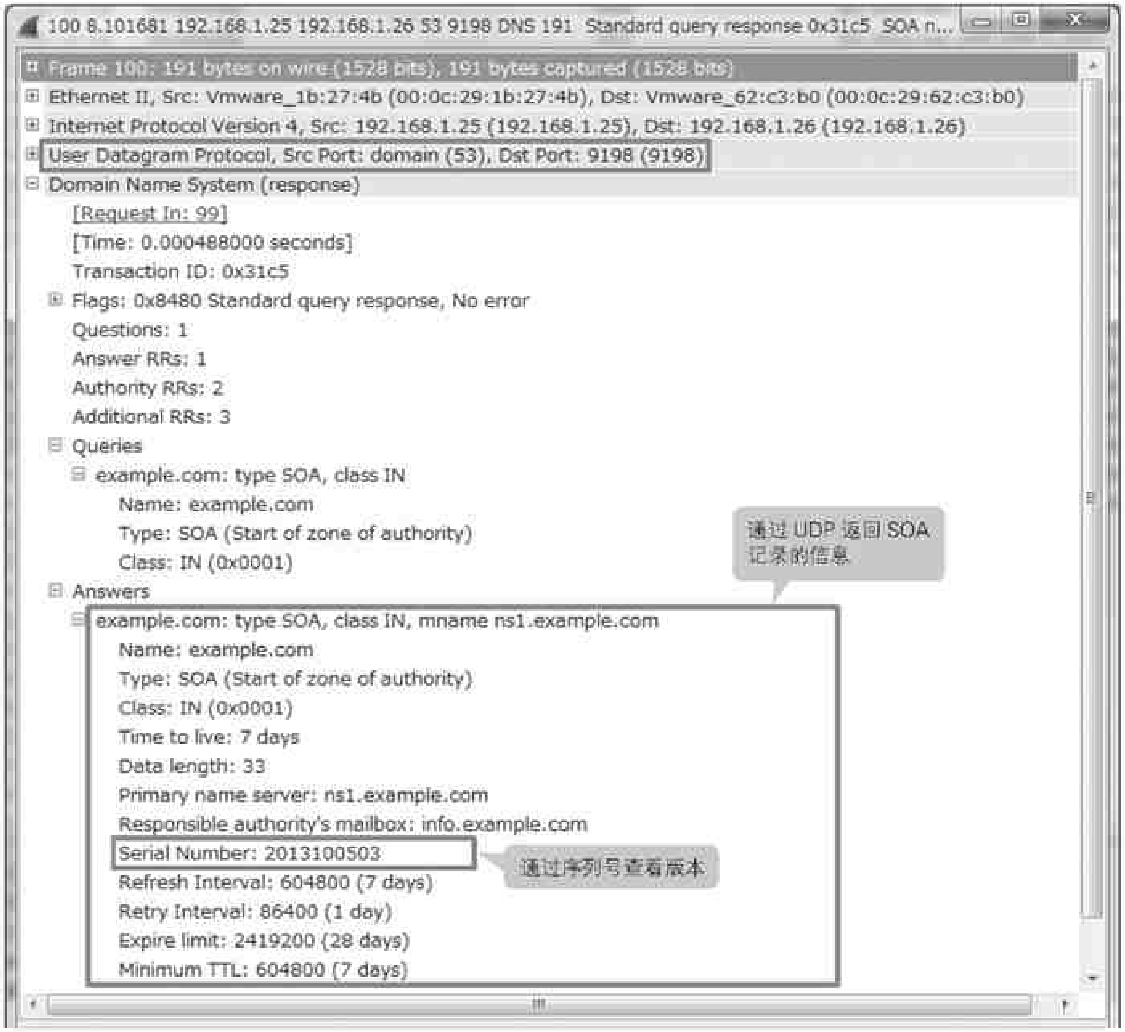


图 **3.2.57**　通过 **UDP** 返回 **SOA** 记录

3 →　从属服务器收到 SOA 记录之后会查看其中的序列号。序列号代表区域文件的版本编号，从属服务器如果发现有新的区域文件（有高于自身版本的区域文件）存在，就会通过 TCP/53 请求主服务器进行区域传输。

4 →　主服务器通过 TCP/53 返回区域信息，区域传输到此结束。

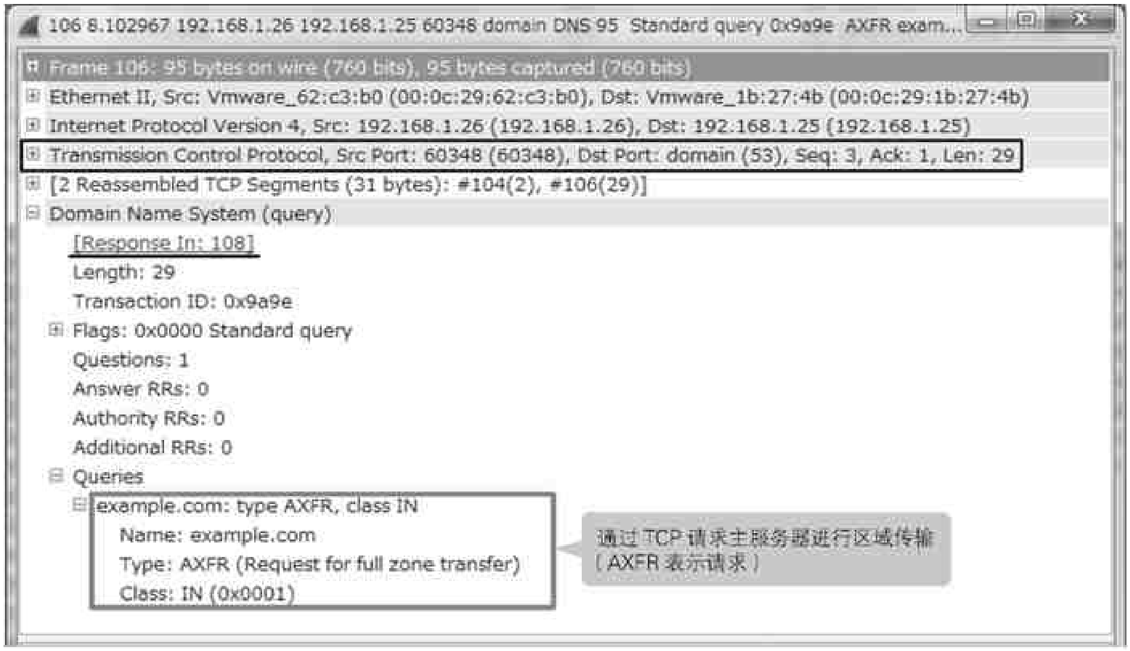


图 **3.2.58**　请求主服务器进行区域传输

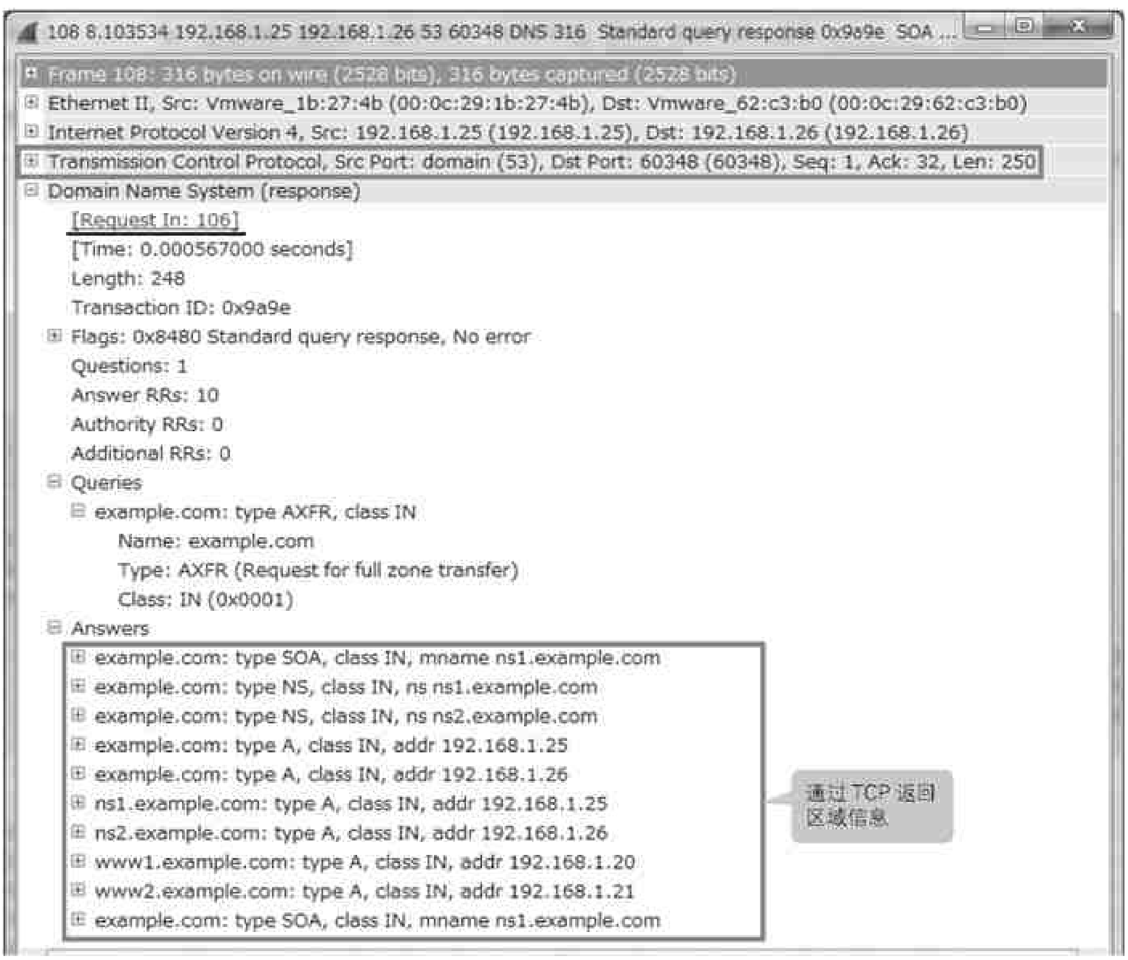


图 **3.2.59**　通过 **TCP** 返回区域信息

3.3　数据安全设计与负载均衡设计

到此为止，我们已经学习了从传输层到应用层的各种技术和设计规格（协议）。接下来，本书将从实用性的角度讲解如何在服务器端使用这些技术，以及我们在设计和架构网站时必须注意的一些事项。

3.3.1　数据安全设计

首先我们来看数据安全设计。在这里，我们要考虑的是如何配置安全区域以及如何确保数据安全。用户和系统管理人员追求的数据安全是在不断变化的，因此我们应该制定出一套容易理解、简明扼要的安全策略，使之无论何时、无论遇到怎样的要求都能够应付自如。

3.3.1.1　整理出真正需要的通信

数据安全设计中最关键的一点是理清通信需求。这里所说的通信需求是指要从哪里（源）向哪里（目的）进行怎样的通信（协议）。我们应整理出这项需求的具体内容，并将它们落实到防火墙的安全策略中去。不同的网站有着不同的通信需求，如果通信需求的数量偏少，不妨参考一下表 3.3.1 中的形式进行统一管理，方便日后查询。

表 **3.3.1**　将通信需求整理成表格，方便日后查询

细分项目		目的				
		互联网	公开服务器 • 公开 Web 服务器 • 外部 DNS 服务器 • 外部邮件服务器 • 外部代理服务器 • NTP 服务器	公司内部服务器 VLAN • 公司内部 Web 服务器 • AD 服务器 • 内部代理服务器 • 内部邮件服务器	公司内部用户 VLAN ①	公司内部用户 VLAN ②
源	互联网	–	HTTP（TCP/80） HTTPS（TCP/443） DNS（UDP/53） SMTP（TCP/25）	×	×	×
	公开服务器 VLAN	HTTP（TCP/80） HTTPS（TCP/443） DNS（UDP/53） SMTP（TCP/25） NTP（UDP/123）	–	SMTP（TCP/25）	×	×
	公司内部服务器 VLAN	×	Proxy（TCP/8080） DNS（UDP/53） SMTP（TCP/25） NTP（UDP/123）	–	×	×
	公司内部用户 VLAN ①	×	×	Proxy（TCP/8080） SMTP（UDP/25） POP（TCP/110） AD 相关（TCP/UDP）	–	ANY
	公司内部用户 VLAN ②	×	×	Proxy（TCP/8080） SMTP（UDP/25） POP（TCP/110） AD 相关（TCP/UDP）	ANY	–

※ 考虑到排版问题，表中内容略有简化。

定义安全区域

安全区域指处于同一数据安全水平的 VLAN 群。实际上这个区域要用到一些彼此之间稍有差异的安全策略，所以并不是完全统一的，但将它们粗略地划分一下会让人更容易理解。我们应根据前面理清的通信需求进一步整理出几个不同的数据安全水平，然后将它们分别定义成不同的区域。设计时将各区域明确定义好，将来就不必做大量的设置修改，管理起来会轻松很多。人们一般将数据安全水平分成三个区域来管理，分别是 Untrust 区域、DMZ 区域和 Trust 区域，下面就针对这三个区域逐一进行讲解。

- **Untrust** 区域

  Untrust 区域位于防火墙外侧，是不可信任的区域。三个区域中数它的数据安全水平最低，不适合配置各种服务器，事实上也绝不能将服务器配置到这里。如果网站需要设置一台在互联网上公开的服务器，那么我们可以认为 Untrust 区域和互联网就是一回事。防火墙是为了防止系统受到来自 Untrust 区域的网络攻击而存在的。

- **DMZ** 区域

  DMZ 区域相当于 Untrust 区 域和 Trust 区域之间的一个缓冲层。它的数据安全水平比 Untrust 区域高，比 Trust 区域低，恰好位于中间地段。人们在 DMZ 区域配置与 Untrust 区域直接进行交互的公开服务器，如公开 Web 服务器、外部 DNS 服务器、代理服务器等。由于不特定的大量用户会来访问这些服务器，所以从数据安全的角度来说，公开服务器是非常危险的，有可能受到形形色色的恶意攻击甚至遭到劫持。不怕一万，就怕万一，为了将不良影响降到最低程度，我们要控制它和 Trust 区域之间的通信。

- **Trust** 区域

  Trust 区域位于防火墙内侧，是可以信任的区域。三个区域中数它的数据安全水平最高，我们应不惜一切代价去保护它。非公开服务器和公司内部用户都配置在这个区域里。

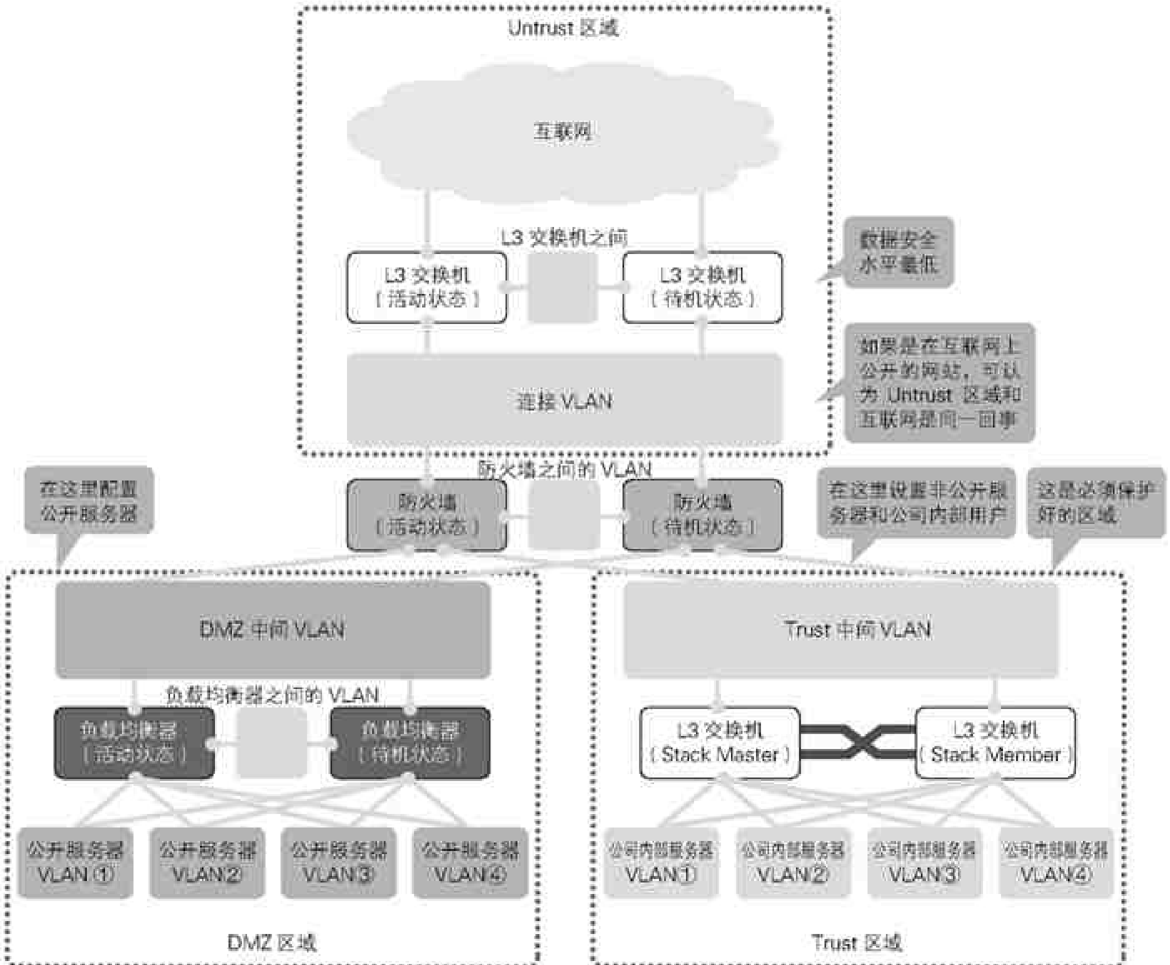


图 **3.3.1**　定义安全区域

将通信要素分组管理

理清需要怎样的通信之后，我们应该提取出其中的要素（IP 地址、网络、协议等），并对这些要素进行分组管理。乍一看这项工作没有任何意义，但实际上它对今后的运行管理起着非常重要的作用。

提取要素并进行分组管理，这听起来也许有些复杂。为了帮助大家理解这一步骤，下面举一个网络设计中十分常见的例子来说明。假设 Trust 区域中有五个公司内部用户 VLAN，而我们需要允许它们通往互联网（Untrust 区域）的所有通信。这时候，一般人都会想到这个办法：先将这五个公司内部用户 VLAN 定义成网络对象，然后制定五个访问控制的策略。但是这个办法有一个缺点，那就是每增加一个用户 VLAN，就要相应地增加一个访问控制策略，这会导致运行管理效率低下。我们

不妨换一个思路，将五个公司内部用户 VLAN 定义成一个组，然后允许该用户 VLAN 组通往互联网的所有通信。这样，即使用户 VLAN 不断增加，我们也只需在该组中添加新的网络对象即可，不必一个个地去制定访问控制的策略。访问控制策略越少就越容易管理，因此，我们应该充分利用分组这个办法来实现高效的运行管理。

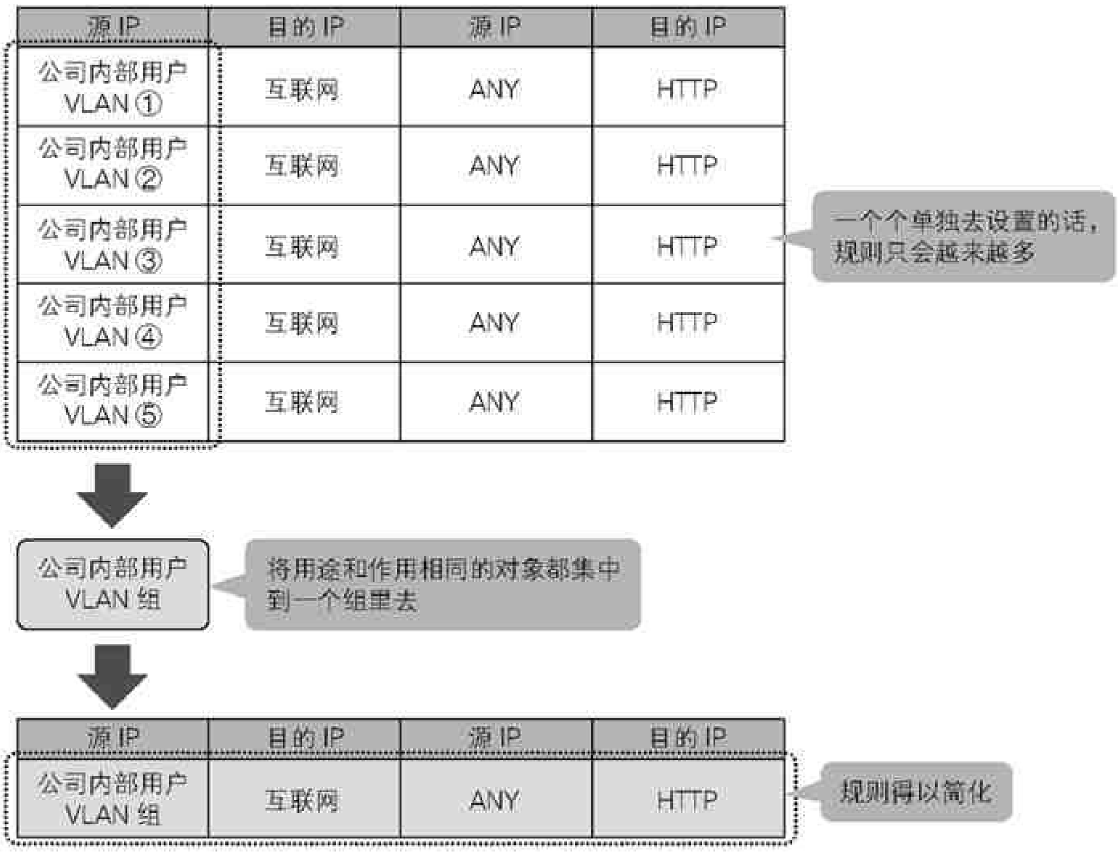


图 **3.3.2**　通过分组管理简化规则

分组管理中有两个容易被忽视的重要环节，那就是设置时要用到的对象名称和组名。我们应预先设计好明白易懂的命名规则，这对今后的管理大有裨益，也能帮助未来接替的管理人员迅速掌握情况。

选择合适的访问控制策略

理清通信需求并分好组之后，我们就要决定合适的访问控制策略了，具体说来就是要决定允许哪些通信、拒绝哪些通信以及允许的话要允许多少、拒绝的话拒绝多少通信。如果只允许最低限度的必需通信当然会很安全，但这种情况的管理会非常麻烦。数据安全水平和运行管理的作业

时间基本上是成正比的，前者越高，后者就越长。因此我们需要综合考虑多方因素，选择整体上比较均衡的策略。

一般说来，如果通信的方向是从数据安全水平较低的区域到较高的区域（例如从 Untrust 区域到 DMZ 区域或从 DMZ 区域到 Trust 区域），我们只能允许最低限度的必需通信通过，如果是相反方向的通信则可以适当地放宽限制。

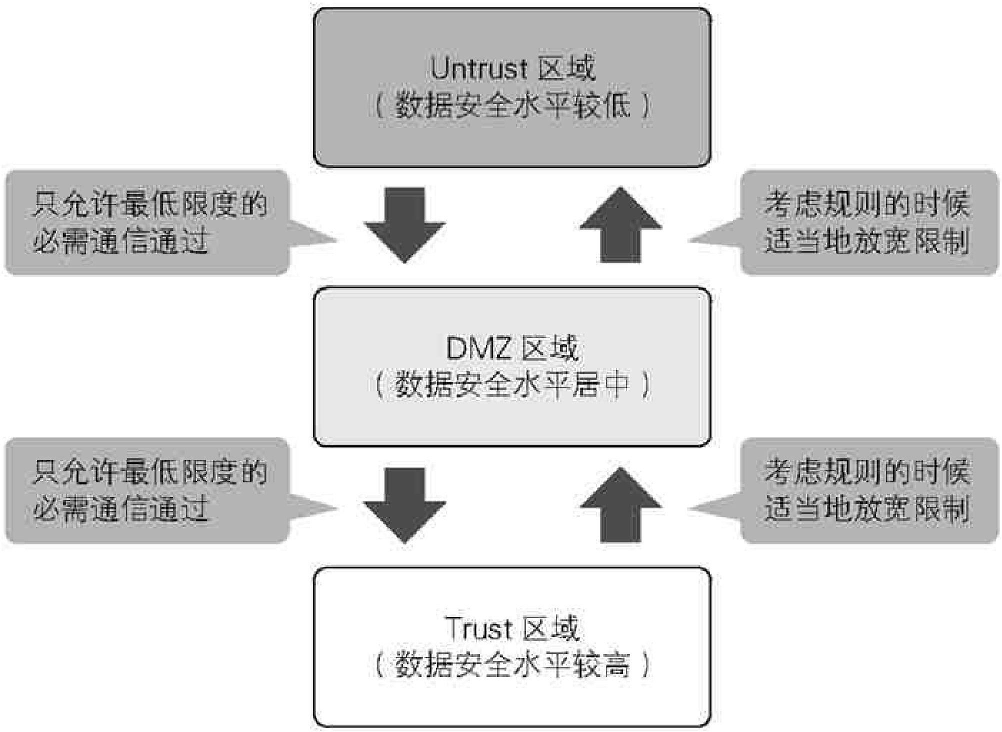


图 **3.3.3**　如果通信的方向是从数据安全水平较低的区域到较高的区域，只能允许最低限度的必需通信通过

3.3.1.2　通过多级防御提高安全系数

最近的防火墙新增了不少功能，如果光是翻看使用手册，简直会让人以为防火墙是无所不能的。不过，防火墙的监控对象毕竟仅限于那些要经过防火墙的通信，人们也不可能将所有的功能都集中在防火墙身上，所以我们应该做的是将防火墙和专用的设备、专用软件搭配使用以形成多级防御，保护系统不受侵害。多级防御是数据安全的根本原则，在这方面我们绝不能马虎行事、掉以轻心。

仔细斟酌需要使用哪些功能

前面已经提到过，最近出现了一种 UTM 形式的新型防火墙，兼具多种功能。然而这种防火墙是一把双刃剑，功能太多以至于性能极其低下。在某些场合，和只作通信控制时的吞吐量相比，这种防火墙在所有功能都被激活时的吞吐量仅为前者的十分之一。无论是什么设备都有自己的强项和弱项，我们不能将所有的功能都交给 UTM 防火墙去实现，而应该找出最适合它去执行的功能，不足之处则让专用的设备和专用软件去弥补，将它们混搭使用以达到博采众家之长的目的。

另外，在选择功能的时候还应该考虑该功能以往的实际使用情况如何。如果不加调查就启用新增的功能，很可能会遇到重大缺陷，最终导致宕机。对于系统来说稳定性是最重要的，所以我们必须谨慎启用新功能，务必在弄清它们的实际使用情况是否值得信任之后再作决定，以免沦为无畏的牺牲品。

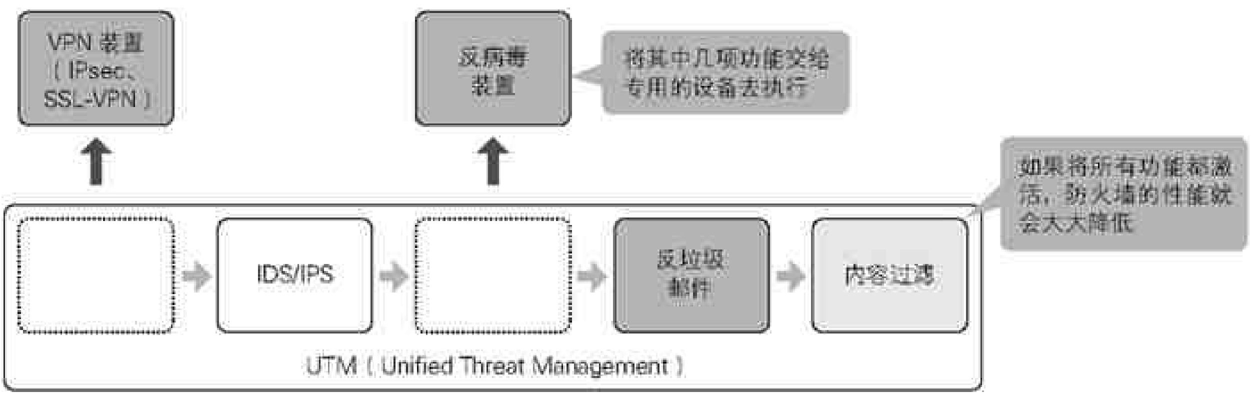


图 **3.3.4**　将不同的功能混搭使用

3.3.1.3　默认启动的服务应控制在最小范围内

网络设备的默认设置会启动很多服务，这可能会导致设备容易受到恶意攻击。例如，在思科公司的交换机和路由器中，管理方面的一些服务是默认为开启的，包括 HTTP 服务和 Telnet 服务等，因此 HTTP 访问和 Telnet 访问都不会受到阻碍，然而这也导致了这些设备易受攻击的一面。为了规避这类风险，我们应将启动的服务控制在所需的最小范围内，保护设备不受恶意攻击。当然，从管理的角度上看有些服务是不可关闭的，遇到那样的情况，我们应该对能够访问的网络进行限制，将可能发生的不良影响控制在最小范围之内。

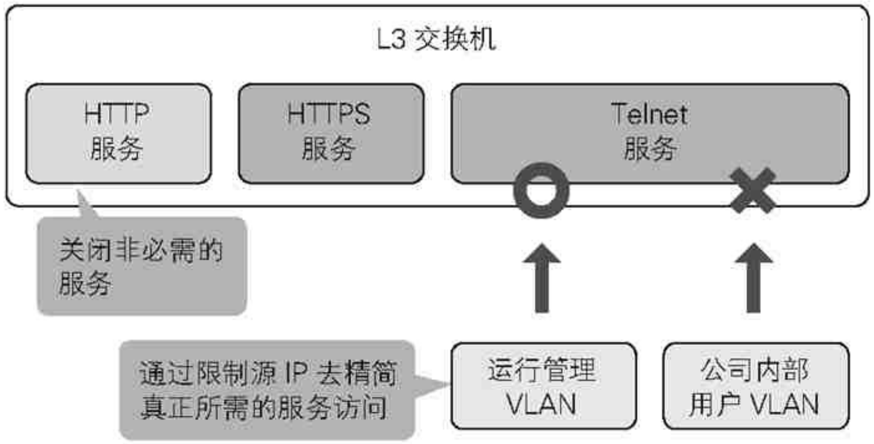


图 **3.3.5**　默认启动的服务应控制在最小范围内

3.3.2　负载均衡设计

下面我们来看负载均衡设计。在这里，我们要考虑的是如何均衡分配应用程序的信息流量负荷，以及在哪个层面上进行该分配。最近人们开发出来的绝大多数应用程序都能够在网络上流通使用，这使得网络信息流量持续猛增。与此同时，通信本身也在变得更为多元化。如何合理地分配日益增多的信息流量、使负荷能够均衡分配，对系统来说是一个非常重要的问题。

3.3.2.1　要高效地均衡负载

负载均衡设计中最关键的一点是理清应用程序层面的通信需求。对于在数据安全设计中整理出来的“从哪里向哪里进行怎样的通信”的具体内容，我们应在负载均衡设计中将它们落实到应用程序的层面上去。

整理出真正需要均衡负载的通信

并不是所有的通信都需要进行负载均衡。首先，我们应该从前面整理好的通信中挑选出真正需要进行负载均衡的对象。如果是在互联网上公开的网站，那么一般要对配置在 DMZ 区域的公开服务器进行负载均衡，通过负载均衡器对互联网上由大量的不特定用户带来的信息流量进行均衡分配。

整理出应用程序层面的通信类型

最近的负载均衡器多为应用交付控制器的形式，应用程序层面的控制比以往更加灵活，应用程序开发人员的要求也变得更加多元化了。下面我们来逐一看下具体的要求。

这里最关键的一点在于是否需要会话保持。负载均衡器的工作原理是将连接分散到多个服务器上，以此来均衡负荷，但是这种分散处理很可能会引起应用程序前后无法呼应的不良后果。所以，我们一定要仔细斟酌是否真的需要会话保持。如果回答是肯定的，那么接下来就要确定使用哪种会话保持以及采用什么时效值。

- 会话保持的种类

  常用的会话保持有两种，分别是源 IP 地址会话保持和 Cookie 会话保持（Insert 模式），这在 3.1.3.2 节中已经介绍过了。源 IP 地址会话保持属于网络层的处理，无需考虑应用程序。Cookie 会话保持则需要执行一个应用程序层面的处理，即 HTTP 报头插入处理，如果选择它来实施会话保持，我们还需要做一些测试以确保该处理对应用程序没有影响。

- 超时时限

  会话超时要比应用程序超时时限稍长才行。太短的话，应用程序在超时之前就被分配到别的服务器上去了，这会导致应用程序前后无法呼应；反过来，太长的话，保持记录不仅毫无意义，还会额外地消耗负载均衡器的资源。所以在时效值这方面，我们也必须仔细确认超时的话设备会有怎样的反应。如果只做几个简单的测试就收场，看到负载均衡和会话保持成功就以为万事大吉，那么到后面很有可能就会吃大苦头。

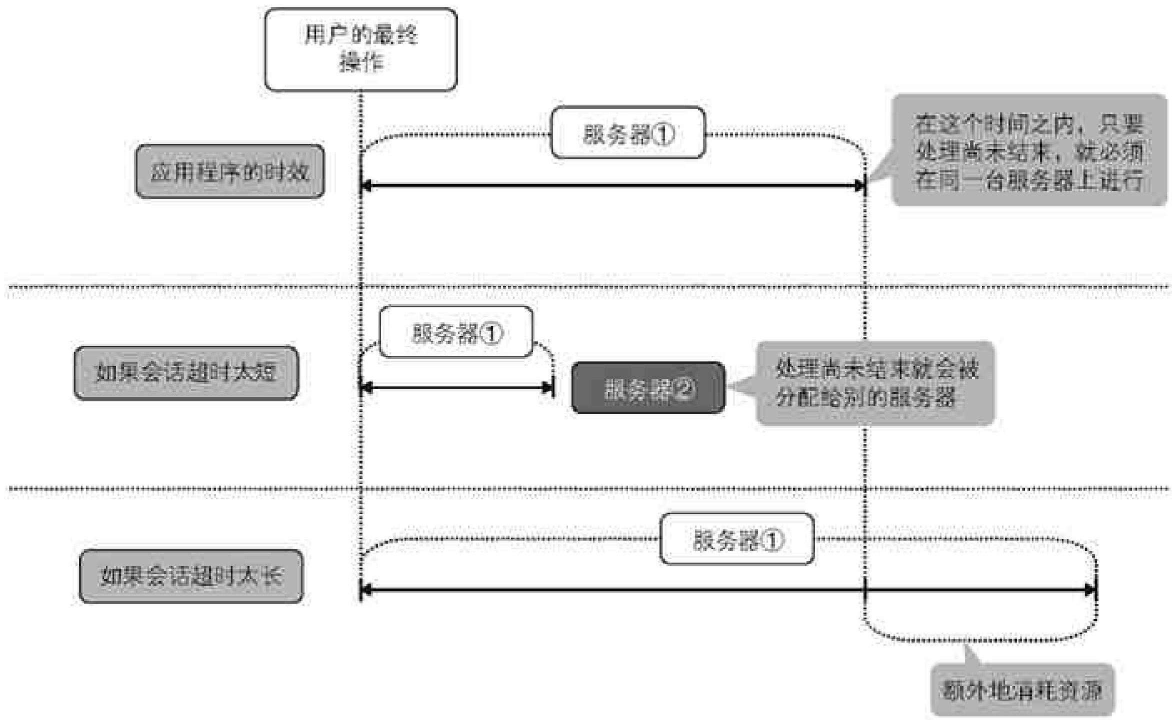


图 **3.3.6**　根据应用程序超时时限设置会话超时

确定将健康检查做到哪个层次

最近的服务器负载均衡环境中有两种不同层次的健康检查，能够帮助我们将不同类别的故障轻松地划分开。例如，HTTP 服务器的健康检查是这样的：通过 L3 检查来检查 IP 地址是否正常，通过 L4 检查或 L7 检查来检查服务或应用程序是否正常。

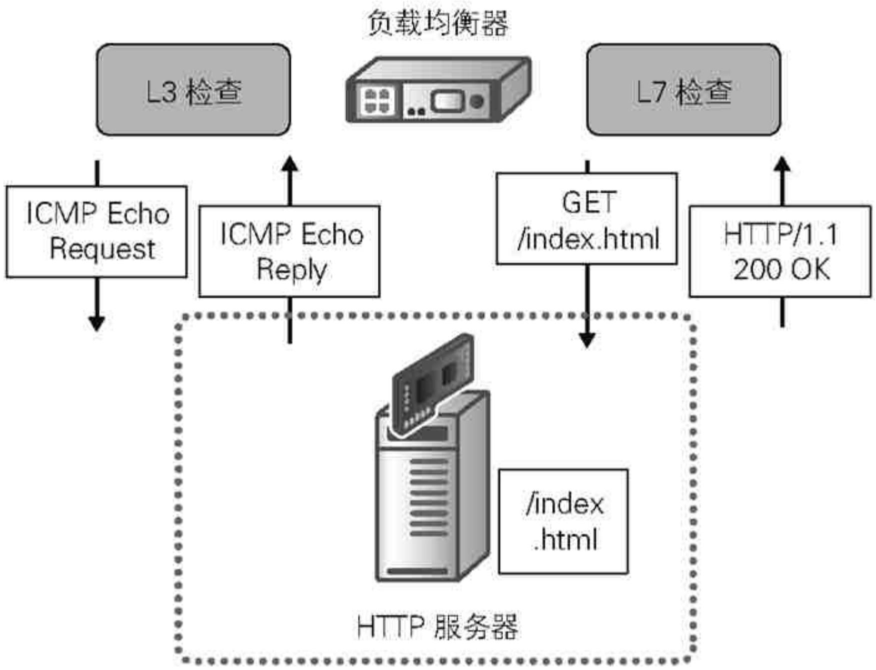


图 **3.3.7**　执行两种不同层次的健康检查

这里需要注意的是，务必对健康检查和服务器负荷进行全面、综合的考量，把握它们之间的平衡。下面我们从间隔和层次这两个角度来具体看一下。

- 健康检查的间隔

  健康检查的间隔越短，检测出故障的速度就越快。但是频率越高，服务器承载的负荷也就越大。我们设计的时候必须要考虑到这一点，让健康检查的间隔不会影响到服务。

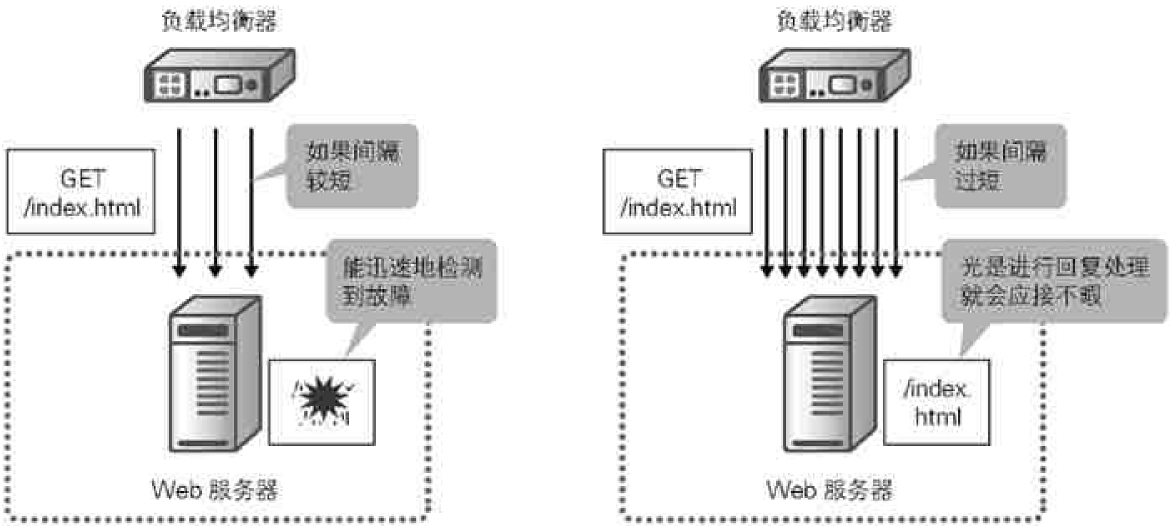


图 **3.3.8**　健康检查的间隔不能太短

- 健康检查的层次

L3 检查只是一种 ICMP 的交互，我们不必将它视为服务器的负荷去考虑。问题在于另一个健康检查——L7 检查。执行 L7 检查能够检测到应用程序层面的故障，但是与此同时，该检查也会增加服务器的负荷。对 HTTPS 的健康检查就是一个很好的例子，HTTPS 需要不断重复执行负荷较大的 SSL 握手处理，很容易成为服务器的一道沉重负荷，如果再加上 L7 检查的负荷，服务器多半会不堪其苦。遇到这种情况，我们可将健康检查的层次降到 L4，尽量减少其对服务的影响。

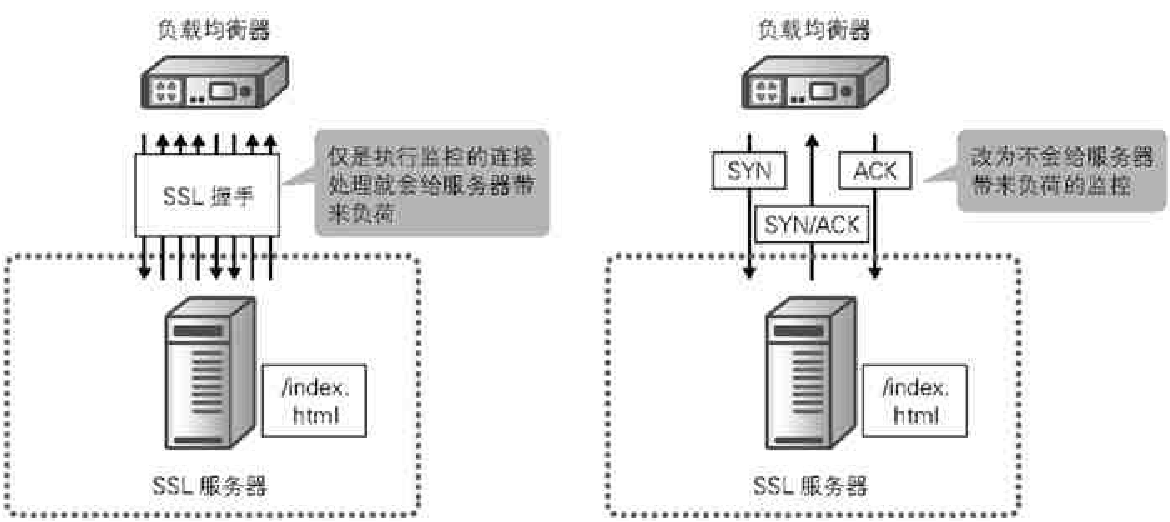


图 **3.3.9**　选择不会给服务器带来沉重负荷的健康检查

由服务器规格和通信类型决定负载均衡方式

负载均衡方式也是非常重要的设计要素之一。负载均衡方式有很多种，各有各的优点和缺点，而且到底哪一种最合适会受到真实服务器环境的影响，所以我们应在综合考虑之后再做选择。不同设备能够使用的负载均衡方式也不同，比较常用的有三种，分别是轮询、加权和比例以及最少连接数。下表中列出了它们各自的优缺点，以供各位读者参考。

表 **3.3.2**　要在理解各方式优缺点的基础上选择负载均衡方式

方式	优点	缺点
轮询（按照顺序分配）	● 分配过程容易理解 ● 如果每个请求的处理时间都一样，就能发挥巨大的作用	● 分配时并不考虑每台负载均衡服务器的规格如何（规格低的服务器可能会不堪重负。因此必须和权重和比例方式同时使用） ● 对于需要会话保持的应用程序无法均等地分配负荷 ● 如果每个请求的处理时间参差不齐，就无法均等地分配负荷
权重和比例（根据权重分配）	● 可根据负载均衡服务器的规格进行分配	● 如果每个请求的处理时间参差不齐，就无法均等地分配负荷
最少连接数（根据连接数分配）	● 可根据负载均衡服务器的规格进行分配 ● 对于需要会话保持的应用程序也能均等地分配负荷 ● 即使每个请求的处理时间参差不齐也能均等地分配负荷	● 如果没有完全理解应用程序的运行过程，就很难明白负载均衡的分配过程

3.3.2.2　启用哪些可选功能

负载均衡器还有不少可选功能，包括 SSL 加速功能、HTTP 压缩功能和连接汇集功能等。在我们的设计中，启用哪些可选功能也是一项非常重要的选择。

准备证书

SSL 加速功能应该是可选功能中最常用的一种功能。启用该功能时，负载均衡器本身即拥有秘钥和数字证书（公钥）。

对于新建的网站，我们应通过负载均衡器生成 CSR，并将其发送给 CA 机构，然后将已被赋予数字签名的数字证书安装到服务器中去。生成 CSR 时要注意密钥的长度，以往密钥长度大多为 1024 位，但出于安全方面的考虑，现在正逐渐向 2048 位转变。不同的 CA 机构会要求不同的密钥长度，所以我们预先要向 CA 机构确认清楚。此外，使用 2048 位时负载均衡器能够处理的 TPS（每秒钟能够处理的 SSL 握手次数）会下降到原来的五分之一左右，因此务必要确认好设备的规格。

如果环境中已有 SSL 服务器存在，那我们就得将安装到 SSL 服务器中的秘钥和证书全部转移到负载均衡器中去。

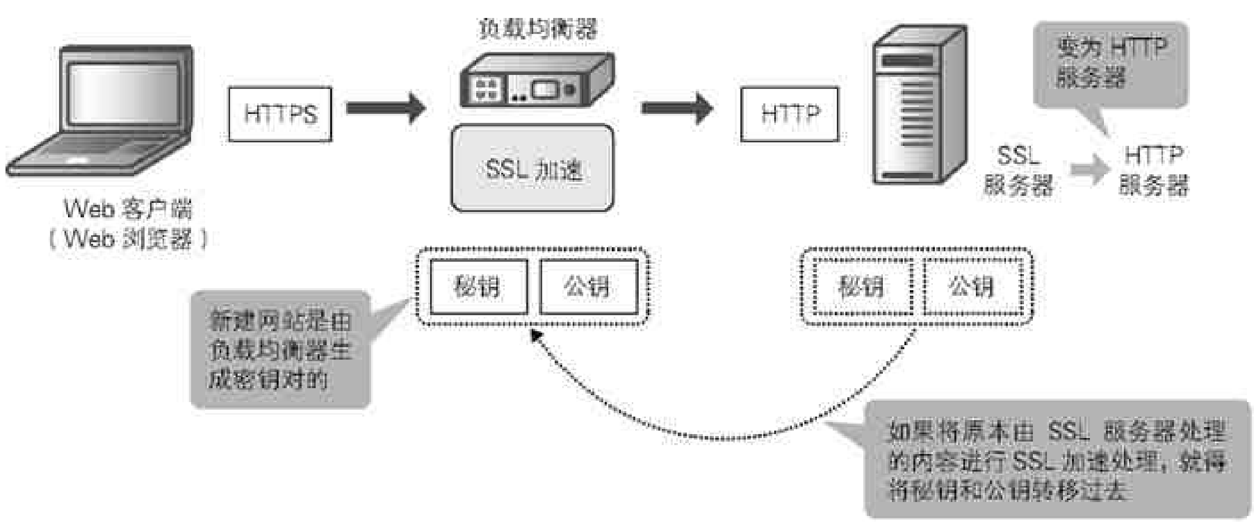


图 **3.3.10**　负载均衡器拥有密钥对

启用 **HTTP** 压缩功能之前要预览效果

对设计人员来说，HTTP 压缩是一项颇难取舍的功能。因为它能发挥多大的效用取决于网站内容，而且并不是在所有网站中都能发挥巨大作用

的，所以无法一概而论。最近出现了一种能够确认压缩效果的网站[10]，在启用压缩功能之前最好能先利用这样的网站预览一下，看看效果如何。

> [10] http://www.gidnetwork.com/tools/gzip-test.php

启用连接汇集功能之前必须谨慎测试

连接汇集功能能够大大减少服务器的 TCP 处理负荷，是一种非常不错的功能。但是它会执行一道比较复杂的应用程序处理，即将来自客户端的应用程序信息流量在负载均衡器内展开还原，然后再交给服务器。因此，当服务器几乎同时收到来自多个用户的应用程序信息流量时，可能会在应用程序上做出一些反常的动作，例如删除报头、修改 IP 地址，等等。考虑到这个负面影响，我们在启用该功能之前必须非常谨慎地进行测试才行。

第 **4** 章　高可用性设计

本章概要

本章将要介绍对于提高服务器端可用性来说所必需的冗余技术和使用该技术时的设计要点，以及各种结构类型中的通信流。

可用性指的是系统少出和不出故障的程度，冗余配置指的是为了保证高可用性而对系统进行多重备份。目前，可以说所有的关键任务系统都处于网络之上。在这样的环境中，即使是一分钟、一秒钟的系统宕机都是足以致命的。从无到有地建立信任关系需要耗费大量的时间，失去它却只要一眨眼的工夫。所以，我们必须设计出合理并且充分的冗余结构以实现高可用性，这样才能避免失去客户的信任。

4.1　冗余技术

从物理层到应用层之中存在着多种多样的冗余技术。在服务器系统中，所有层的所有关键设备都需要无一遗漏地进行冗余配置，本书将从中选出与网络相关的冗余技术以及设计时必须注意的事项，逐一分层解说。

4.1.1　物理层的冗余技术

物理层的冗余技术是通过将多个物理要素集结成一个逻辑要素的形式来实现的。听起来似乎很高深，但大家不用想得太复杂。这句话的意思就是，不管物理要素有多少，我们只需把它们当成一个整体去看待就好了。

这些要素中，本书将着重解说链接、设备和网卡这三种物理要素，中间会穿插一些其他相关的内容。

4.1.1.1　将多条物理链路集结成一条逻辑链路

将多条物理链路集结成一条逻辑链路，这叫作链路聚合。用思科公司的术语来说叫作“以太通道”，用惠普公司和 F5 Networks 公司的术语来说叫作“端口汇聚”（trunk），实际上都是同一个意思。链路聚合是一种让链路的带宽扩展和链路本身的备份能够同时实现的技术。

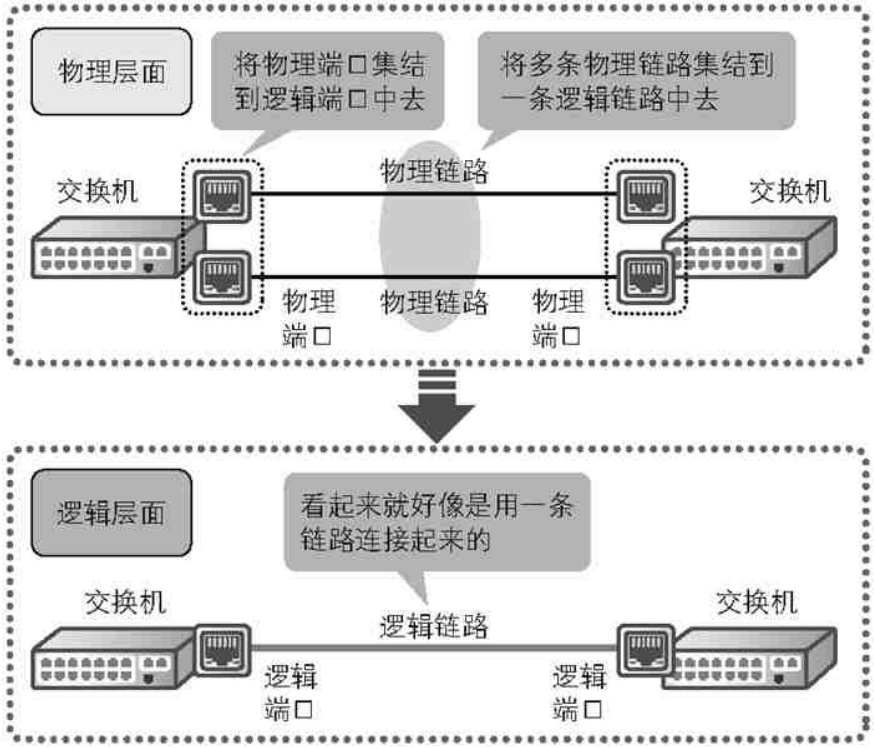


图 **4.1.1**　将多条物理链路集结成一条逻辑链路

链路聚合将交换机的多个物理端口捆绑在一起形成一个逻辑端口，让这个端口和另一台交换机的逻辑端口连接，以此来生成逻辑链路。正常情况下，聚合起来的多条物理链路就好像一条链路那样工作，能够确保所有物理链路所需的带宽。当链路发生故障时，故障链路会被断开，切换成冗余应对。因故障引起的宕机时间从 ping 的层面上看只有短短的一秒钟，因此，对应用程序层面几乎是没有任何影响的。

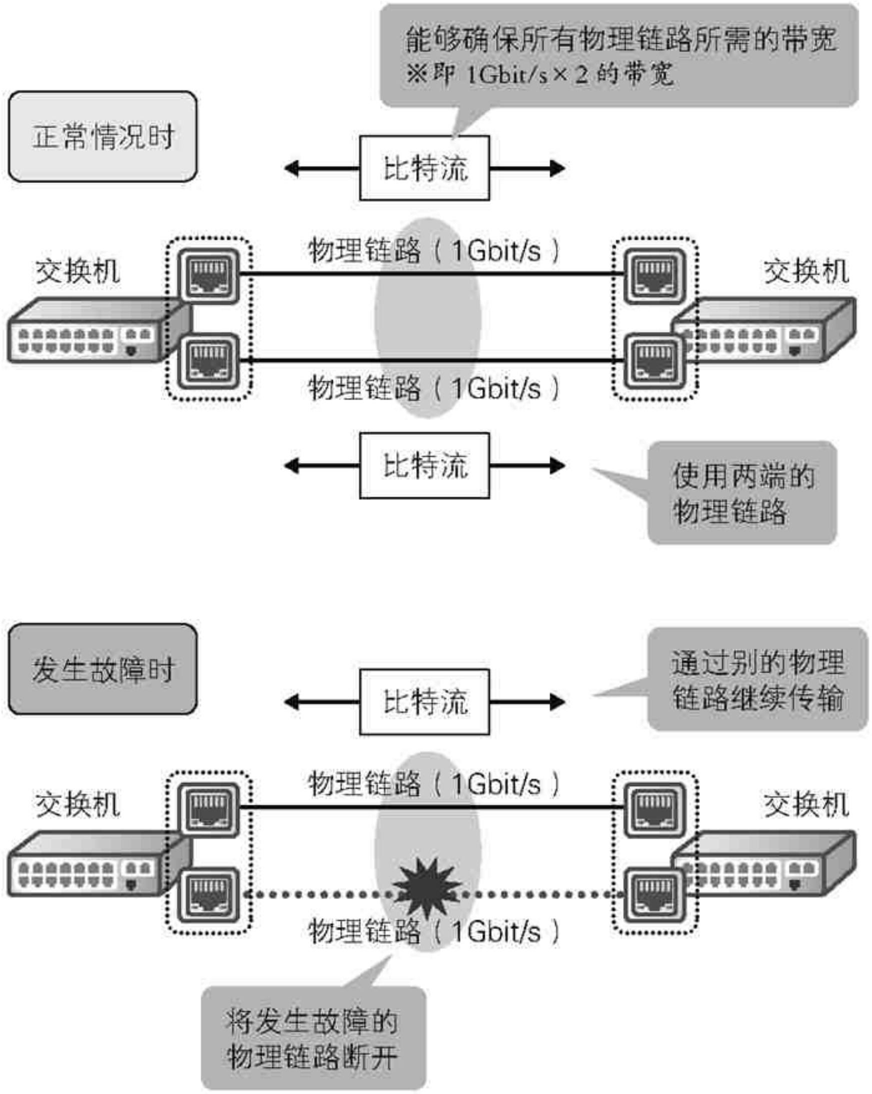


图 **4.1.2**　通过链路聚合同时时间带宽扩展和链路备份

模式大致可分为三种

链路聚合大致可分为三种模式，分别是静态、PAgP（Port Aggregation Protocol，端口聚合协议）和 LACP（Link Aggregation Control Protocol，链路聚合控制协议）模式，它们彼此之间互不兼容，设计时我们只能选择其中一种使用。

- 静态

  静态是将链路无条件地集结到链路聚合中，从而形成一条逻辑链路的模式。它不需要用到太多的协议，在三种模式中是最简单易懂

的。如果选择静态模式，那么两端的设备必须都设置成静态模式才行。

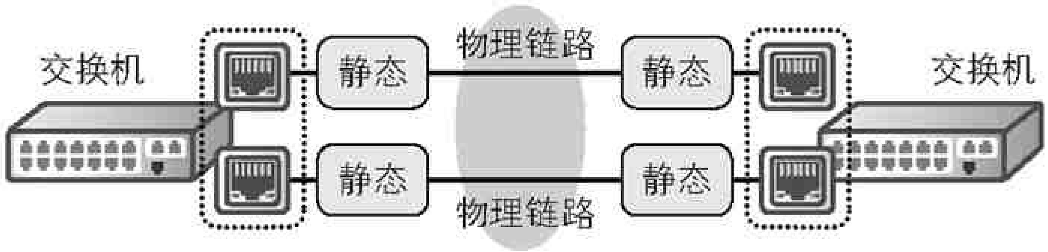


图 **4.1.3**　选择静态时必须将两端设备都设置成静态模式

- **PAgP**

PAgP 是一种用来自动生成链路聚合的协议。它是思科公司专有的协议，如果服务器环境全部采用思科公司的设备，那么最好选择它。PAgP 向对方发出征询的试探包之后，逻辑链路就会自动生成。

PAgP 有两种工作模式，一种是协商（Desirable）模式，另一种则是自动（Auto）模式。打个形象的比方来说，协商模式相当于肉食动物，自动模式则相当于草食动物。协商模式主动向对方发送 PAgP 包，积极地要求建立逻辑链路；自动模式则是被动地接收对方发来的 PAgP 包，然后才建立逻辑链路。

在进行普通的网络设计时，为了统一设置，人们往往会将两端设备都设置成协商模式。

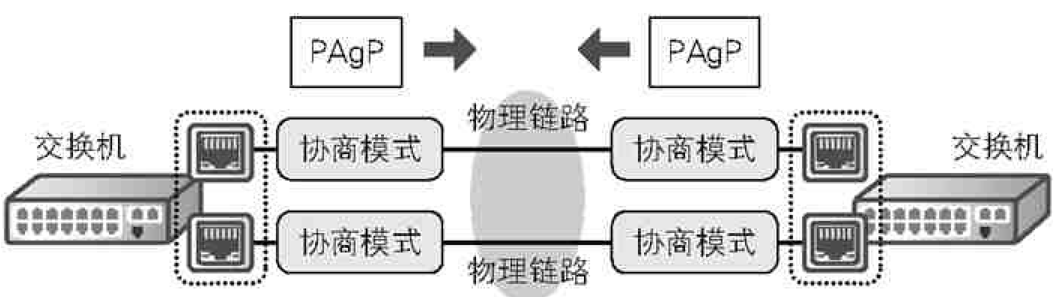


图 **4.1.4**　选择 **PAgP** 时一般会设置成协商模式

- **LACP**

LACP 也是一种用来自动生成链路聚合的协议，不过它是基于 IEEE802.3ad 进行标准化的，能够用在由多家供应商提供的不同品牌的设备组成的网络环境中。LACP 向对方发出征询的试探包之后，逻辑链路就会自动生成。

LACP 有两种工作模式，一种是主动（Active）模式，另一种则是被动（Passive）模式，它们分别相当于 PAgP 的协商模式和自动模式。主动模式主动向对方发送 LACP 包，积极地要求建立逻辑链路；被动模式则是被动地接收对方发来的 LACP 包，然后才建立逻辑链路。

在进行普通的网络设计时，为了统一设置，人们往往会将两端设备都设置成主动模式。

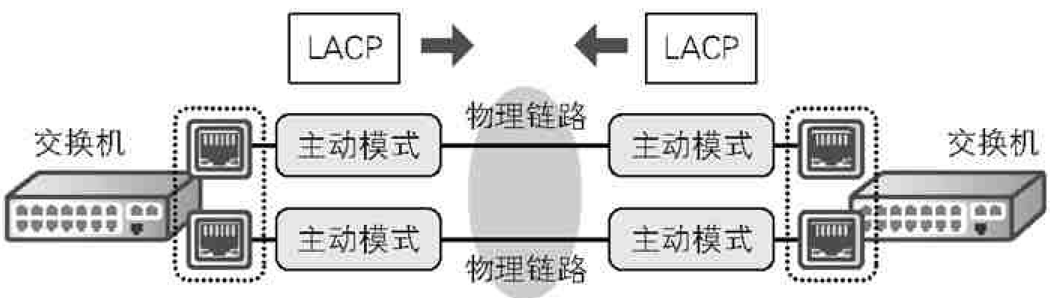


图 **4.1.5**　选择 **LACP** 时一般会设置成主动模式

负载均衡方式非常重要

说到链路聚合中的扩展带宽，其原理不过是将负荷分散到各条物理链路中去而已。这样，在实际传输帧的时候，每一条物理链路都会承担一部分负荷，于是从全局来看就实现了带宽扩展。这里的关键在于采用什么负载均衡方式，如果错误地选择了不合适的负载均衡算法，物理链路的负荷就会有所偏重，从而达不到均衡分配的效果。

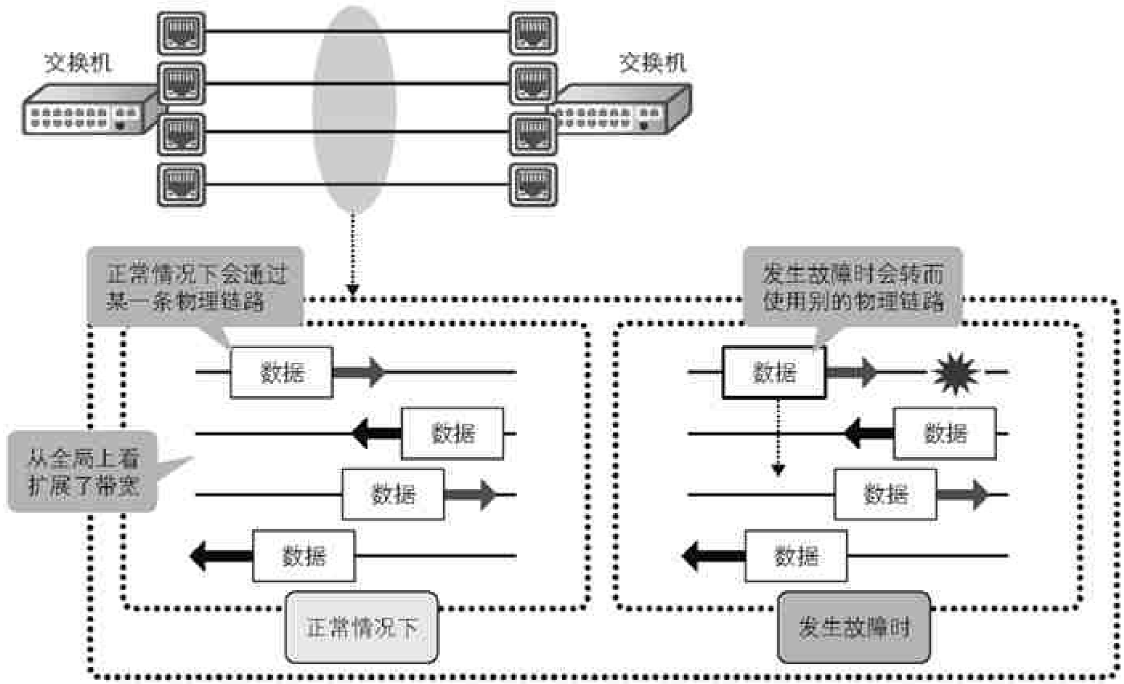


图 **4.1.6**　通过负载均衡在全局上实现带宽扩展

我们来看一个具体的例子吧。思科公司的 Catalyst2960/3750 系列设备默认的负载均衡方式是使用源 MAC 地址。如果我们基于源 MAC 地址去分散负荷，那么来自不同 VLAN 的通信就会大量集中到一条物理链路上。为什么会这样呢？因为来自不同 VLAN 的通信的源 MAC 地址一定是默认网关的 MAC 地址，因此只会用到一条物理链路，导致通信效率低下。

可能的话，最好将负载均衡安排在传输层（源 / 目的端口号）或者网络层（源 / 目的 IP 地址），这样负载均衡的效率会比较高。采用哪一种负载均衡方式应该视具体的设备而定，所以我们一定要预先确认好设备的规格。

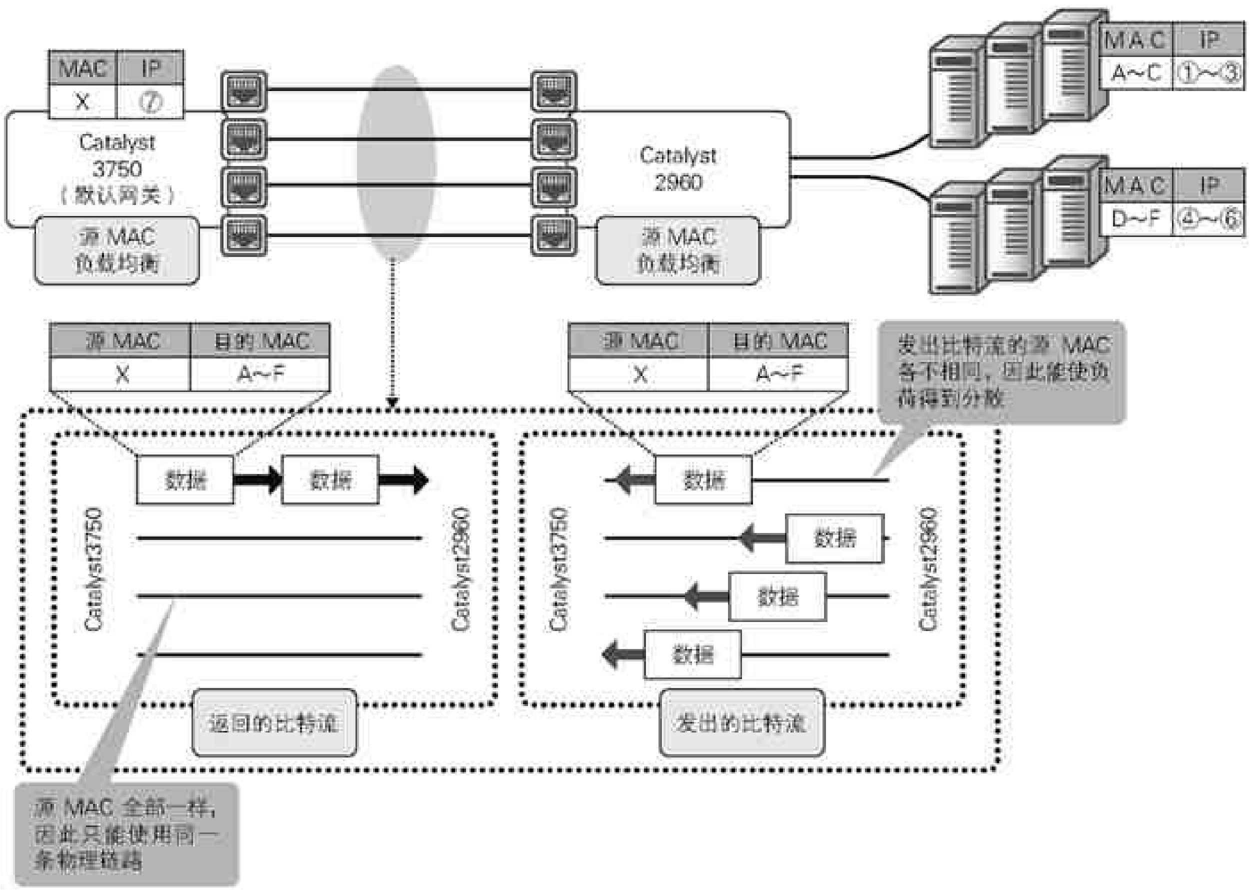


图 **4.1.7**　基于源 **MAC** 地址分散负荷会使通信有所偏重，导致负载无法均衡分配

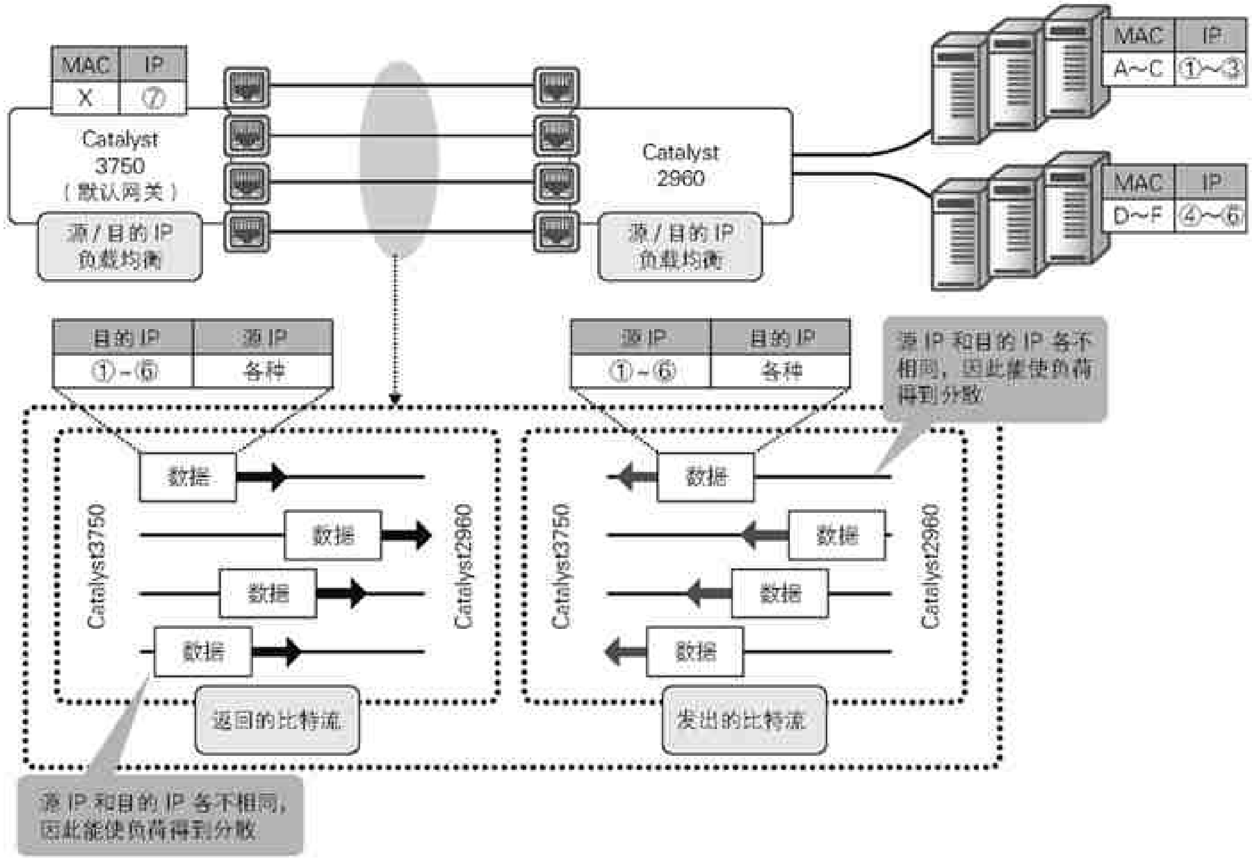


图 **4.1.8**　改变负载均衡方式以扩展带宽

4.1.1.2　将多个物理网卡集结成一个逻辑网卡

将多个物理网卡集结成一个逻辑网卡，这叫作网卡组合，用 Linux 的术语来说叫作网卡绑定，我们可以认为二者是同一个意思。网卡组合是一种让带宽扩展和网卡备份能够同时实现的技术，由于它是对服务器网卡进行的设置，乍一看和网络似乎没有什么关系，但实际上它和网络的冗余设计息息相关，适当了解一下是有利无弊的。本书将按物理环境和虚拟环境分别介绍几种比较常用的网卡组合方式，。

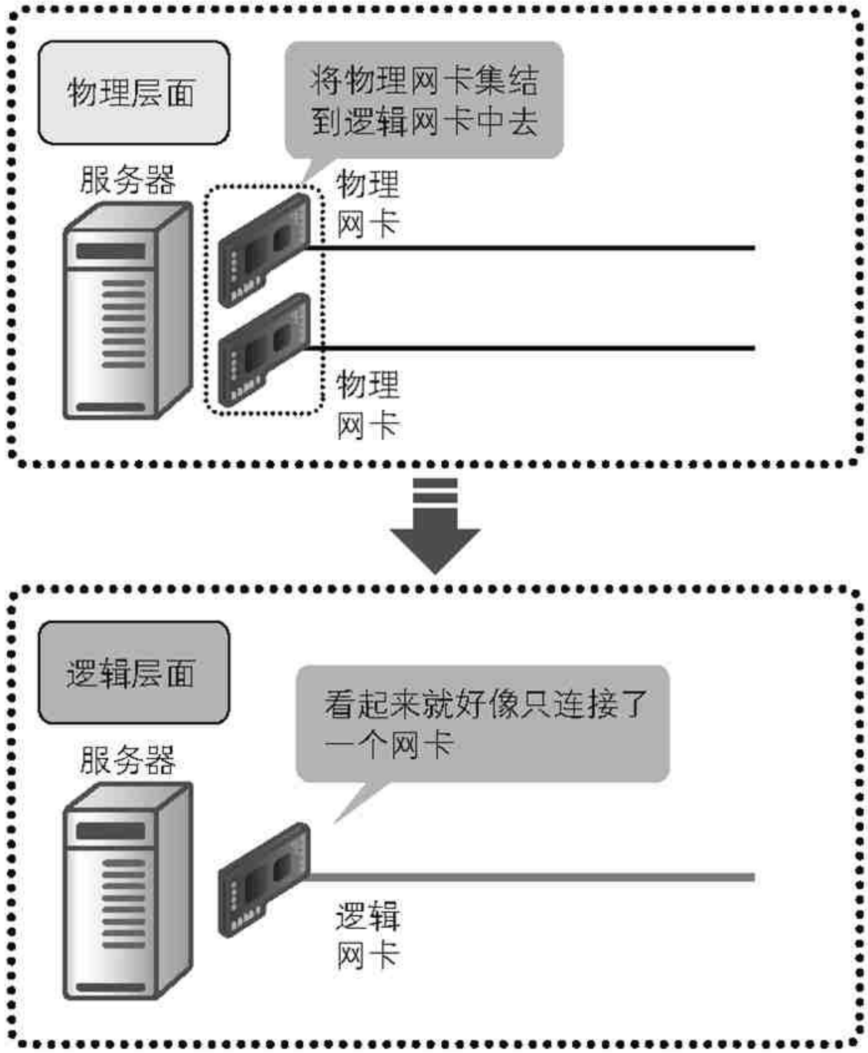


图 **4.1.9**　将多个物理网卡集结成一个逻辑网卡

掌握物理环境的三种网卡组合方式

物理环境中的网卡组合是通过软件设置出来的。说到比较知名的网卡组合软件，那就要数 Intel PROSet 和 BACS/BASP 了。网卡组合设置好以后，一个新的逻辑网卡就建成了，我们就可以在这个新的逻辑网卡上进行下一步的动作。另外，组合网卡的时候我们必须指定使用哪一种组合方式。

在物理环境中可以使用的网卡组合方式有很多种，其中比较常用的有三种，分别为 AFT（Adapter Fault Tolerance，适配器容错）、ALB（Adapter Load Balancing，适配器负载均衡）和链路聚合。这里，我们以通过 Intel PROSet 软件设置网卡组合为例来看一下这三种组合方

式。

- **AFT**

  AFT 是对物理网卡进行冗余配置的一种模式。它在服务器上装配有两个网卡，一个是活动网卡，另一个则是待机网卡。正常情况下仅使用活动网卡，当活动网卡发生故障时执行故障转移，由待机网卡顶替活动网卡工作。由于一般情况下 AFT 仅使用活动网卡，所以两个物理网卡的通信量是完全分离开的，一旦活动网卡的处理量达到最大极限，就无法再接受更多的通信。不过 AFT 即使发生故障也容易排除，加上运行管理简便易行，因此容易获得系统管理人员的青睐。

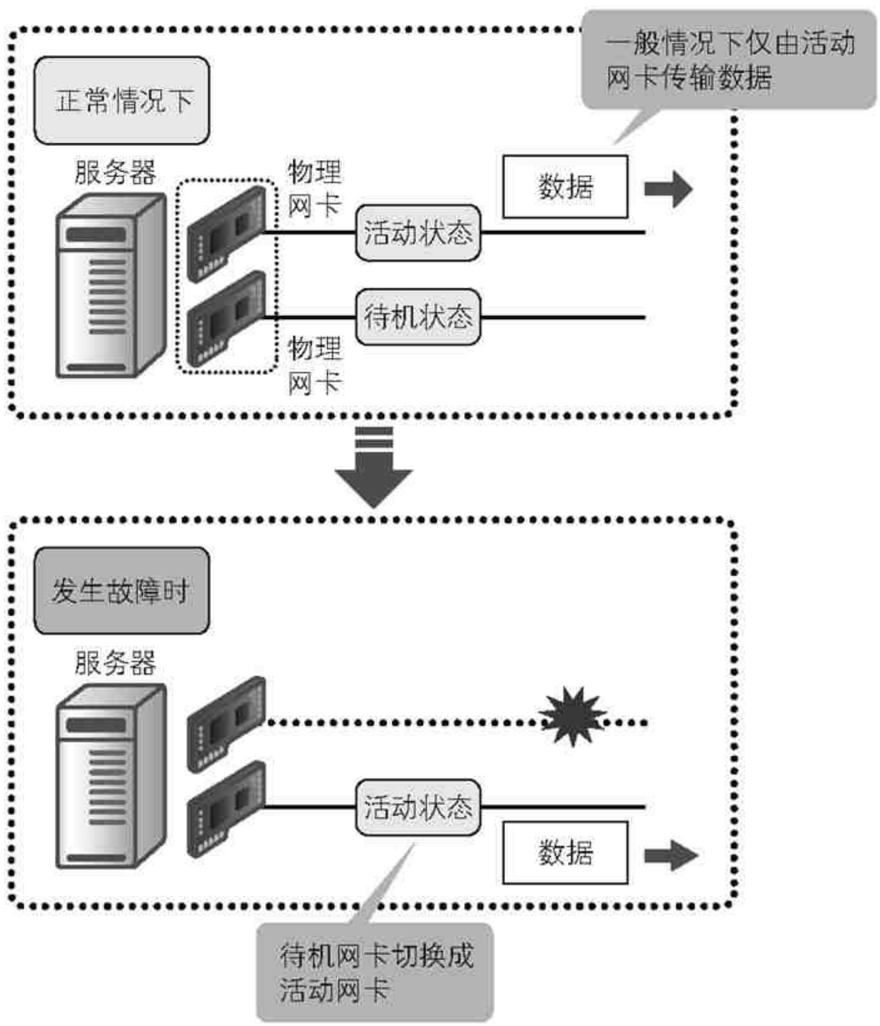


图 **4.1.10　AFT** 由活动网卡和待机网卡构成

- **ALB**

  ALB 是让物理网卡的冗余配置和带宽扩展同时实现的一种模式。它同样是在服务器上装配两个网卡，一个是活动网卡，另一个是待机网卡 [1]，不过一般情况下两个网卡都会用到。如果其中一个网卡发生故障，就通过另一个网卡维持通信。ALB 在一般情况下就使用两个网卡，所以通信效率要比 AFT 高。然而它有一个设计上的缺点——由于两个网卡都必须连接到同一台交换机上才行，所以一旦连接的那台交换机发生故障，通信就会中断 [2]。

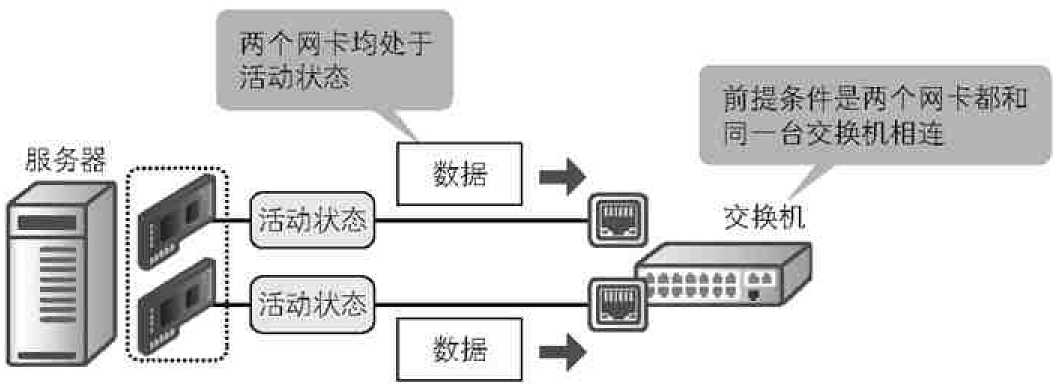


图 **4.1.11　ALB** 也由活动网卡和待机网卡构成

- 链路聚合

  前面我们介绍过物理链路的链路聚合，这里的链路聚合可以理解为是网卡版本的链路聚合，它能让带宽扩展和物理网卡备份同时实现。一般情况下，该模式会根据一定的方式去选择通信的物理网卡并扩展带宽。如果同组中的某个网卡发生故障，就会立刻切换成另一个网卡，确保通信正常进行。链路聚合在一般情况下就使用所有物理网卡，因此通信效率较高。然而它有一个缺点，那就是连接的交换机一旦发生故障就无法通信了 [3]。另外，由于交换机也需要设置链路聚合，所以服务器负责人员和网络负责人员应就采用哪种协议和负载均衡方式进行充分的讨论和商议。

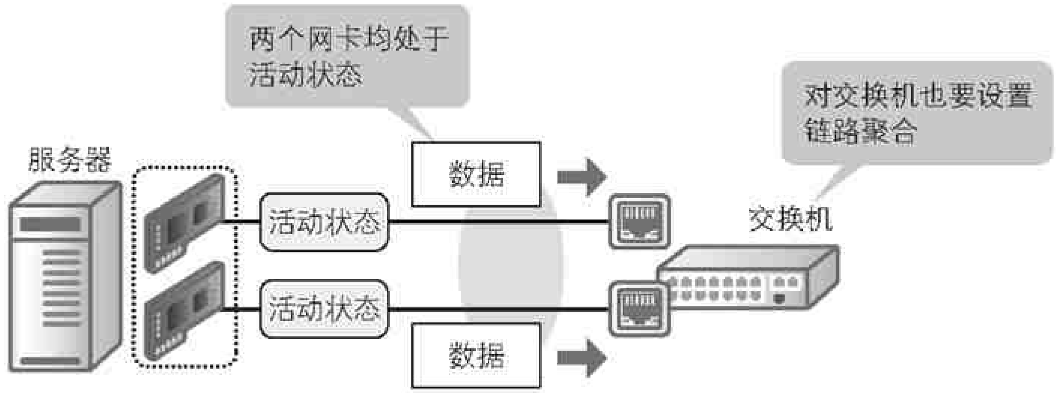


图 **4.1.12**　交换机也需要设置链路聚合

[1] 实际上并非所有通信都会交给活动 / 待机网卡去处理。只有发往不同 VLAN 的数据包才会被分配给双网卡（基于目的 IP 地址分配），其他数据包（广播数据包、多播数据包以及发往同一 VLAN 或是同一 VLAN 收到的数据包）则是由早已定义好的主网卡去处理的。

[2] 对于这种情况，我们需要利用 StackWise 技术或 VSS 架构出一台逻辑交换机，并将连接的物理交换机分开。关于 StackWise 技术和 VSS 将 4.1.1.3 节中详细说明。

[3] 对于这种情况，我们需要利用 StackWise 技术或 VSS 将几台交换机集结成一台逻辑交换机，并将连接的物理换机分开。

在虚拟交换机中设置虚拟环境的网卡组合

虚拟环境中的网卡组合是在虚拟软件超级管理程序上的虚拟交换机中设置的。虚拟机通过虚拟交换机和物理环境连接，将物理网卡关联到虚拟交换机上，然后通过该物理网卡建立网卡组合。虚拟环境的网卡组合有两个要点，分别是故障检测和负载均衡方式。

- 故障检测

  故障检测指的是根据什么信息来检测故障。例如，VMware 中有两种故障检测机制，一种叫作链路状态检测，另一种叫作信标检测，二者之间的差异在于它们作用的层不同。

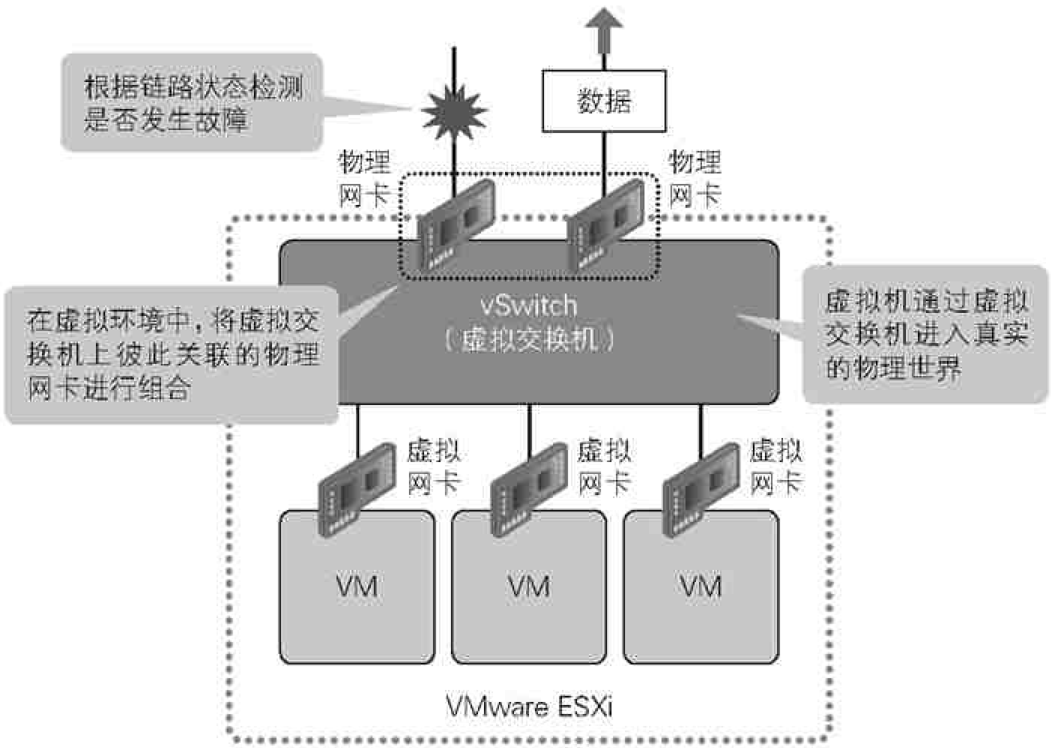


图 **4.1.13**　根据链路状态检测是否发生故障

链路状态检测是在物理层对链路状态（连接还是断开）进行检测，信标检测则是在数据链路层周期性地发送特殊的帧 [4] 并确认其丢失情况，以此来进行检测。二者之间，笔者推荐采用前者。这是因为采用信标检测时，信标帧可能会被其他设备误判为非法帧，而且有时该机制检测不出正在发生的故障，这点需要多加注意。

- 负载均衡方式

虚拟环境中的网卡组合同样是一种通过将通信分散到各个物理网卡上，使带宽扩展和网卡备份能够同时实现的技术。负载均衡方式指的是采用哪些物理网卡。VMware 中有四种负载均衡方式，分别为明确的故障转移、基于端口 ID 的负载均衡、基于源 MAC 哈希的负载均衡和基于 IP 哈希的负载均衡。其中最为常用的是明确的故障转移和基于端口 ID 的负载均衡方式。

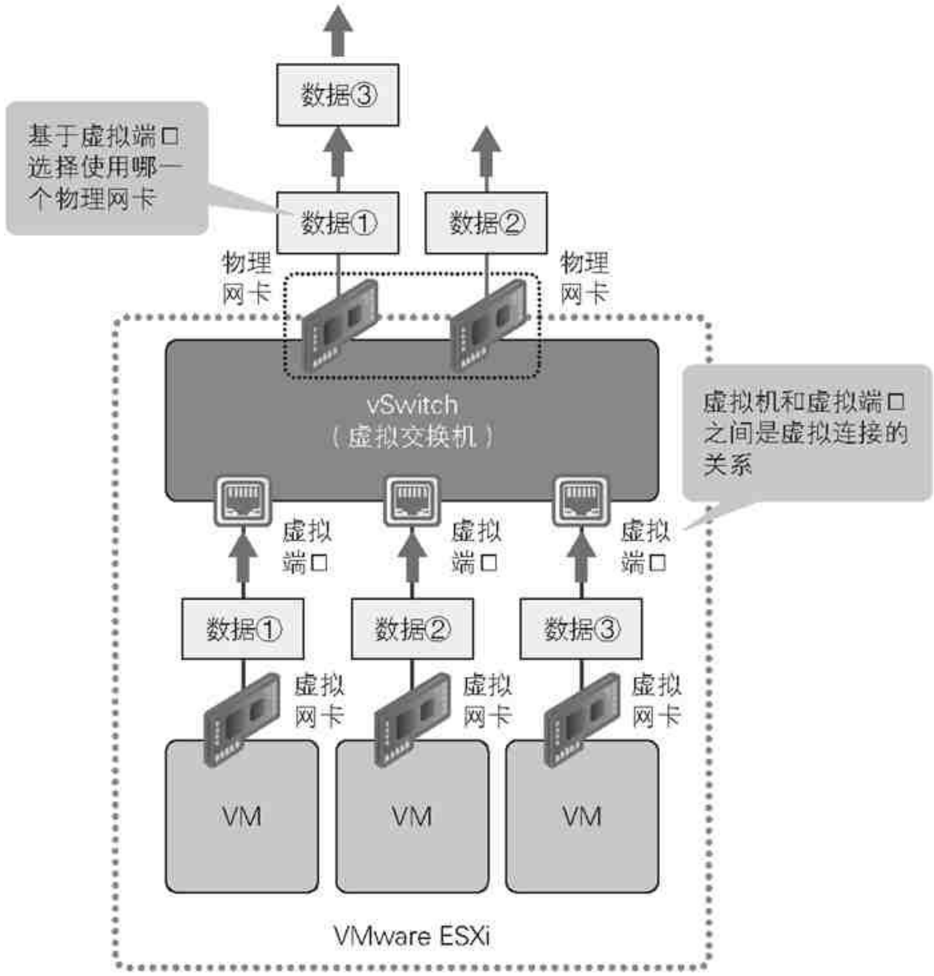


图 **4.1.14**　基于虚拟端口的端口 **ID** 选择使用的物理网卡

明确的故障转移是指在服务器上装配两个网卡，一个是活动网卡，另一个则是待机网卡。正常情况下仅使用活动网卡，当活动网卡发生故障时执行故障转移，由待机网卡顶替活动网卡工作。

基于端口 ID 的负载均衡则是一种以端口为单位切换使用网卡的负载均衡方式。这里所说的端口不是 TCP 或 UDP 的端口号，而是虚拟机所连接的虚拟交换机的虚拟端口。为每个虚拟端口号（端口 ID）选择与之对应的物理网卡，就能够实现全局意义上的负载均衡。

[4] 该帧的源 MAC 地址为物理网卡的 MAC 地址，目的 MAC 地址为广播 MAC 地址，以太网类型为 0x05ff。

将不同种类的物理网卡组合使用

我们在组合网卡时还需要充分考虑物理网卡的配置问题。用于服务器的物理网卡大致可分为两类，一类是安装在主板上的板载网卡，另一类则是添加在扩展插槽中的扩展网卡。我们应尽量将这两类网卡组合使用以提高冗余效果。假如我们只选择四端口的扩展网卡绑定成组，那么扩展插槽一旦坏掉，通信就会中断。所以，为了尽量减少因物理构成要素发生故障而造成的影响，我们应将扩展网卡和板载网卡组合使用。

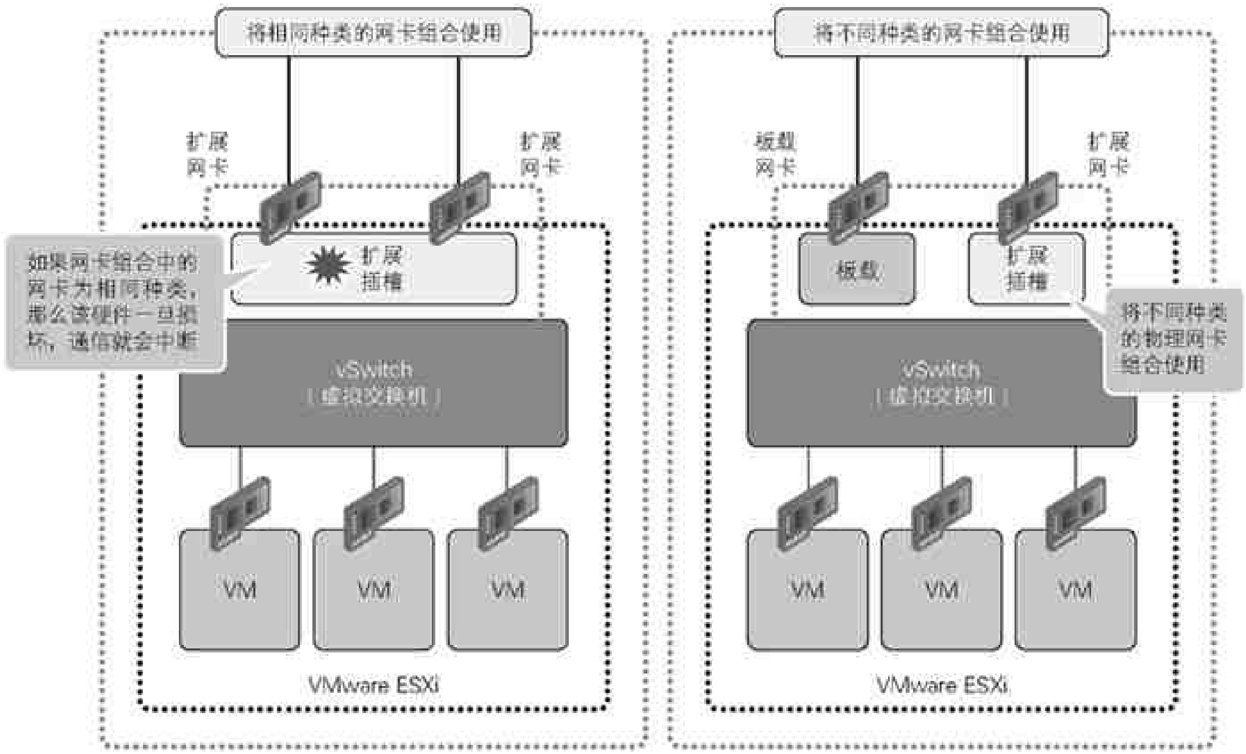


图 **4.1.15**　将不同种类的网卡组合使用

务必将连接的对象分开

我们在组合网卡时还需要考虑物理网卡的连接对象。如果将物理网卡都连接到同一台物理交换机上，那么该交换机一旦损坏我们就无力回天了。所以，应将两个物理网卡分别连接到两台不同的物理交换机上，这样才能够保证其中一台物理交换机发生故障时，另一台仍然可用。

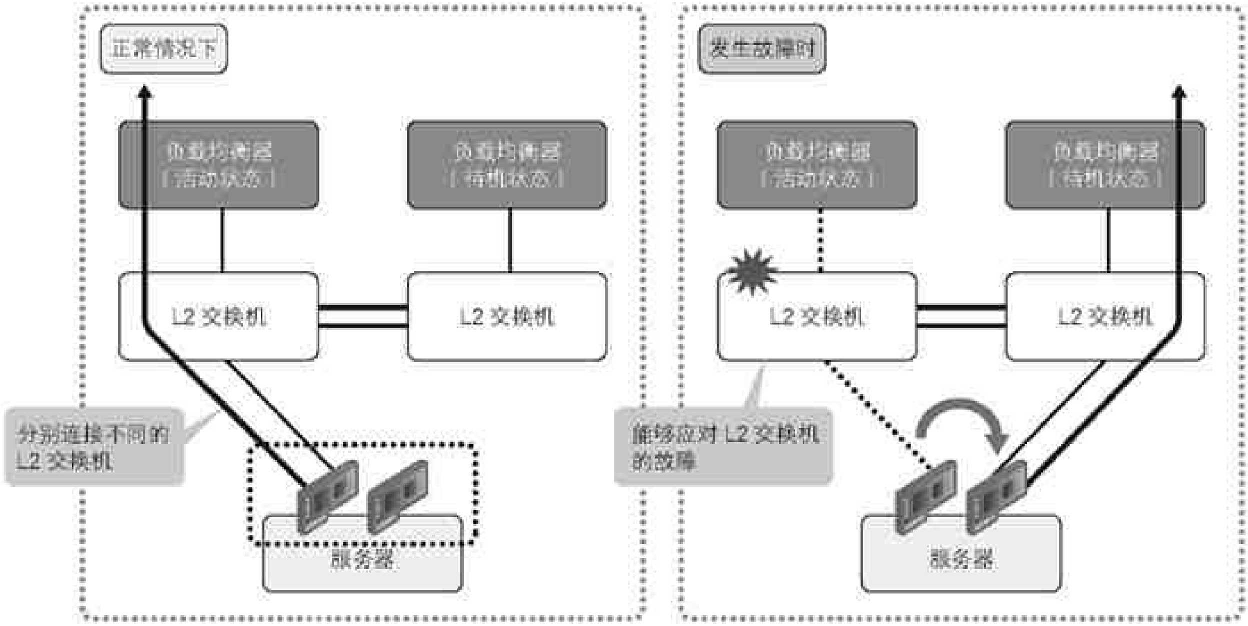


图 **4.1.16**　分别连接不同的物理交换机

刀片服务器让物理结构得以简化

刀片服务器是将名为“刀片”的薄型服务器插入名为“刀片机箱”的外壳中使用的一种服务器[5]。一般来说，服务器数量较多时布线作业就会非常麻烦，刀片服务器则能够大大地简化布线作业并提高机架集中率，从而使设备物理层面的运行管理工作变得轻松。如今，刀片服务器正在逐渐成为系统架构的必备要素之一。

> [5] 有些生产商可能会使用不同的称呼，本书选择使用比较常用的称呼来进行讲解。

当我们将刀片从刀片机箱的正面插入时，装在刀片机箱背面的扩展模块就会自动完成内部的连接布线。那些连接结构我们是看不到的，因此，刀片服务器刚刚问世时人们常常半信半疑，担心它是否真的完成了连接布线，不过事实证明那些担心都是多余的。

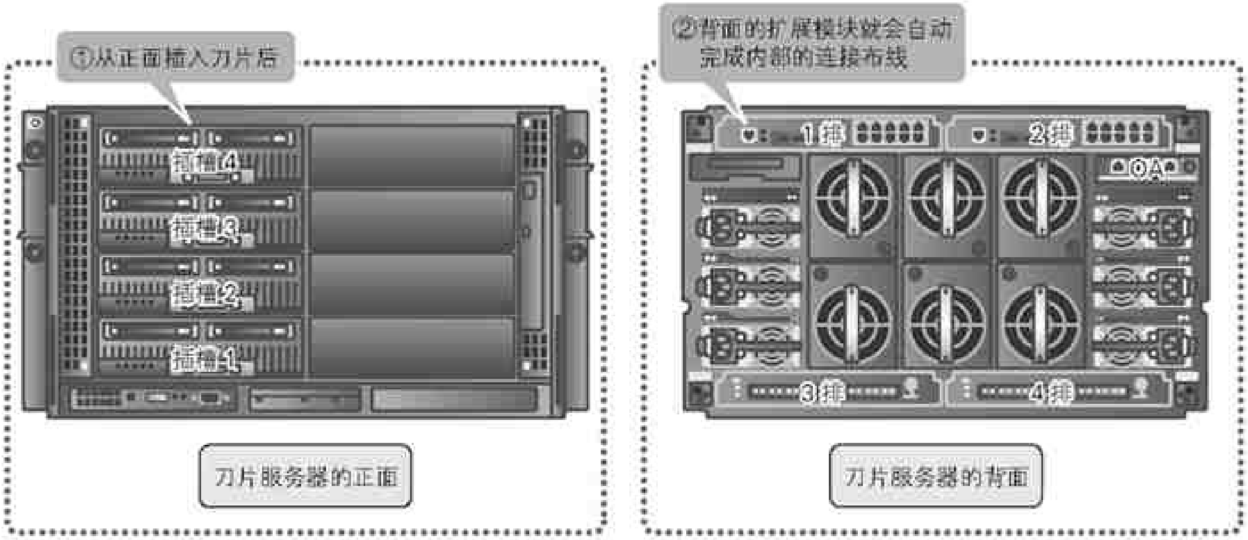


图 **4.1.17**　刀片服务器内部会自动连接布线

扩展模块有很多种类。种类不同，网络结构就不同，负责维护的工程师也就不同。下面，本书将选择网络工程师负责范围内的、比较常用的交换机模块来进行介绍。

- 交换机模块

  交换机模块是怎么一回事呢？想象一下刀片机箱中藏着一台交换机，你就能心领神会了。前面已经讲过，我们将刀片插入刀片机箱后，机箱内部会自动完成连接布线的工作。刀片的插入位置（插槽）和安装在刀片中的夹层卡[6]决定了接线的位置。举例来说，如果将刀片插入插槽 1，它就会连接到各个交换机模块的 1 号端口；将刀片插入插槽 2，它就会连接到各个交换机模块的 2 号端口。原本露在外面的线缆全部在机箱内部彼此相接，因此也不再需要进行与服务器之间的布线工作。需要布线的地方仅剩一处，那就是通过外部端口与外部交换机连接的那个部分。

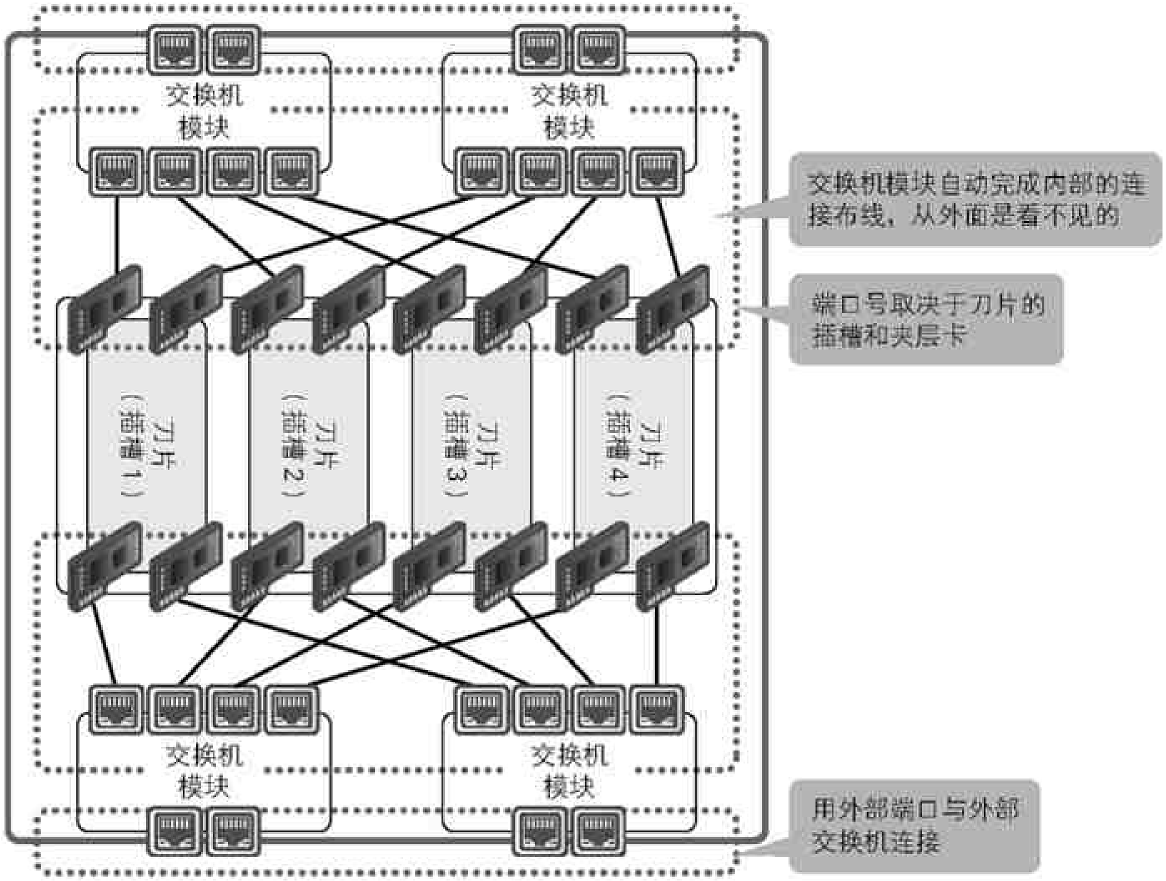


图 **4.1.18**　连接端口取决于刀片插入的插槽

我们在使用交换机模块的时候，必须注意它和管理模块之间的关系。刀片服务器的整体管理是通过一个叫作管理模块的模块实现的，它在 IBM 刀片服务器中心被称为 AMM（Advanced Management Module，高级管理模块），在惠普刀片服务器系统中则被称为 OA（Onboard Administrator，板载管理器）。交换机本身也受管理模块的管理，在默认状态下其 IP 地址和主机名都是由管理模块设置的。此外，对交换机的管理访问也必须经过管理模块才行，有些设计还规定了它必须使用不同于服务器的 VLAN。这些规格上的细节我们都必须仔细确认。当然，我们也可以将交换机模块设置成不受管理模块管理的形式，如果架构工作全部由网络工程师来完成，那么我们应将它排除到管理模块的管理范围之外。

[6] 指插入刀片的主板来使用的扩展卡。

4.1.1.3　将多台物理设备集结成一台逻辑设备

最近，将多台物理设备集结成一台逻辑设备的技术获得了急速的普及。这项技术能够一举解决网络中存在的多个问题，包括冗余配置、增强传输能力、无环回、简化结构，等等。目前，它已在高可用设计中占据了无可替代的一席之地。

这种类型的冗余技术分为好几种，针对不同的设备应采用不同的技术。下面，本书将要介绍可以用于思科公司 Catalyst3750 系列的 StackWise 技术和可以用于 Catalyst6500/6800 系列的 VSS（Virtual Switching System，虚拟交换系统）的设计要点。

StackWise 技术

StackWise 技术是一项可以用于 Catalyst3750 系列和刀片交换机的冗余技术。它使用特殊的堆叠线，能够连接的交换机多达九台，进而将这些交换机集结成一台巨大的逻辑交换机。从物理层面来看存在着多台交换机，然而从逻辑层面来看就像只有一台交换机在工作。因此，系统管理人员需要管理的 IP 地址和各种设置信息等设计要点也都只有一个。

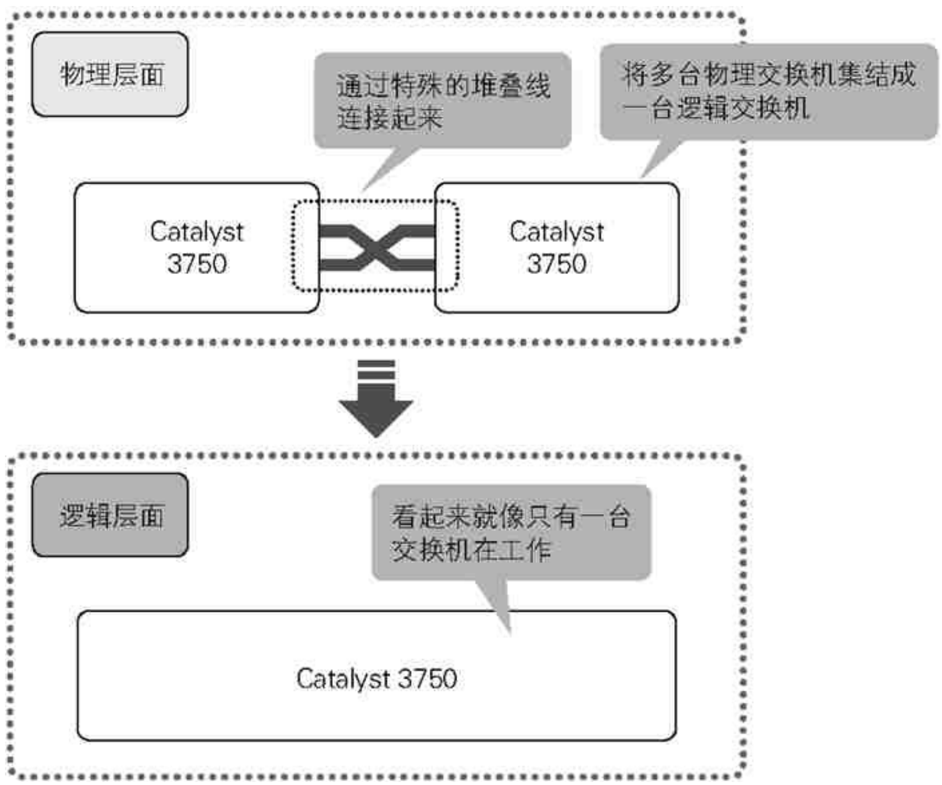


图 **4.1.19**　将多台物理交换机集结成一台逻辑交换机

- 选出主交换机

形成堆叠的交换机由一台负责全局管控的主交换机和几台分交换机构成。主交换机负责处理单播和多播的路由选择，以及复制各分交换机的设置信息和传输信息（FIB 表[7]）等工作，在堆叠中扮演着最重要的角色。StackWise 技术让主交换机集中执行一些复杂的处理（路由选择协议处理等），让分交换机分头执行一些简单的处理（传输处理等），进而使所有的交换机各司其职、各施其长。

主交换机可以根据几个条件筛选出来。不过，我们一般需要预先设置好优先值，以保证能够筛选出特定的交换机。优先值默认为 1，最大为 15，拥有最高优先值的交换机将当选为主交换机。在设置好主交换机的优先值之后，我们不妨进一步设置好下一讲将要成为主交换机的候补对象的优先值，这样，发生故障的时候就不至于手忙脚乱了。

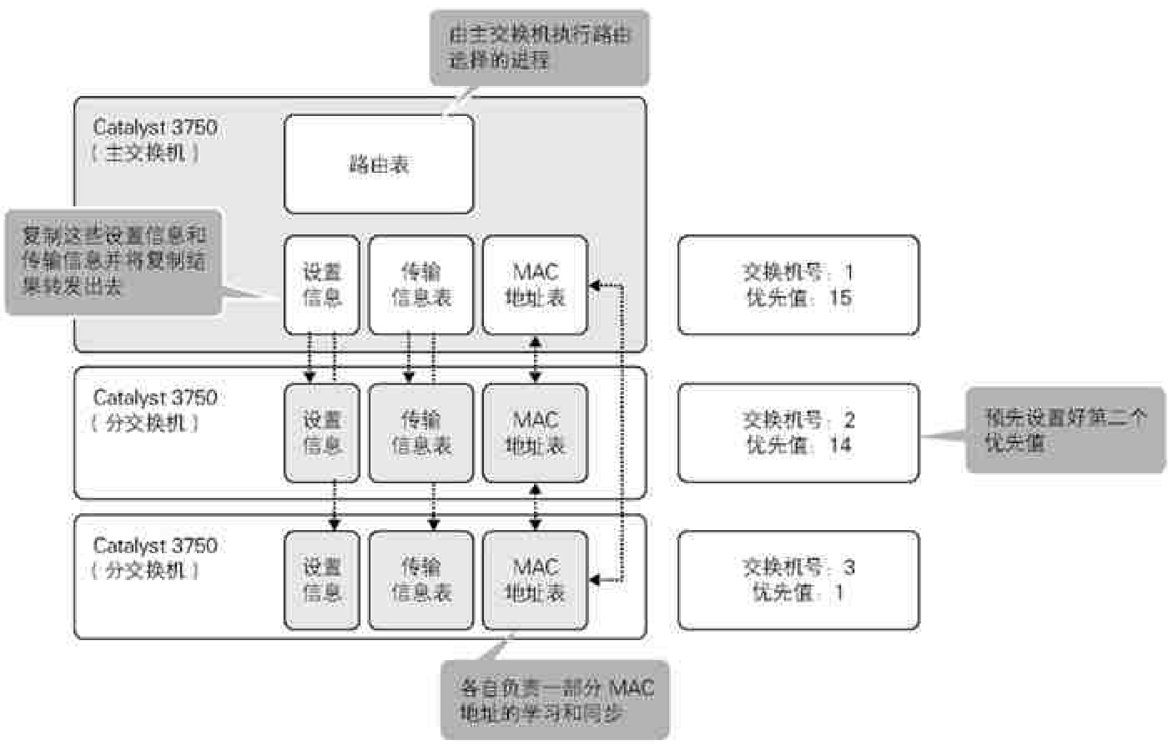


图 **4.1.20**　主交换机负责全局管控

关于主交换机我们还必须了解一点——原本拥有最高优先值的主交换机发生故障又恢复正常之后，并不会自动升级为主交换机。如果需要将其恢复为主交换机，我们必须通过重启当前的主交换机等操

作以再次执行筛选进程。

- **MAC** 地址

我们在使用 StackWise 技术的时候还需要注意 MAC 地址。形成堆叠结构时，堆叠的 MAC 地址（堆叠 MAC 地址）默认为主交换机的 MAC 地址，这在正常情况下是没有问题。然而主交换机一旦宕机，问题就随之出现了——在特定的环境（LACP 和 STP 并用的环境）下，为了应对宕机现象而进行的 MAC 地址切换处理会导致通信中断。考虑到这种情况，无论是在怎样的环境下，我们都应提前输入命令 stack-mac persist timer 0。这样，即使主交换机宕机，它的 MAC 地址也能够在新的主交换机中继续使用，就不会发生不必要的通信中断了。

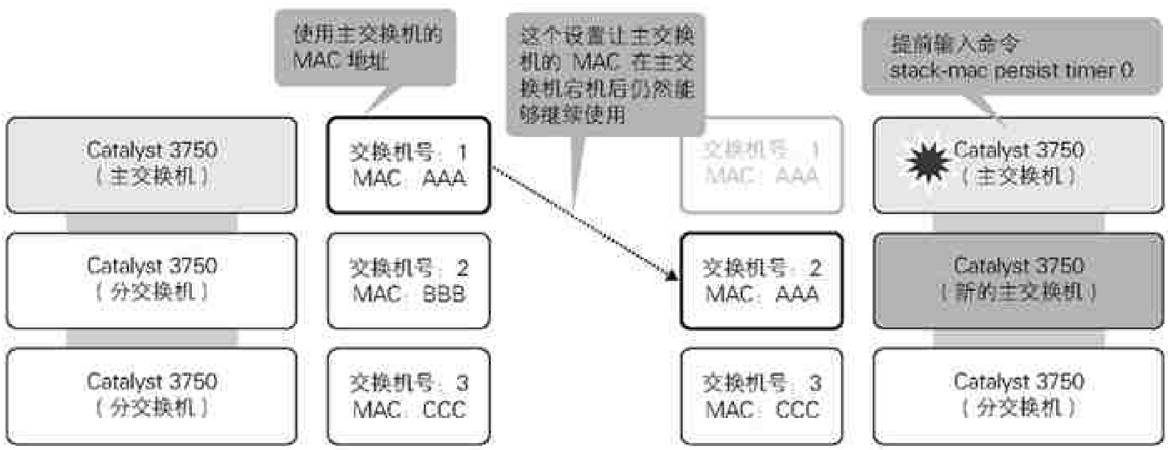


图 **4.1.21**　通过 **stack-mac persist 0** 命令使 **MAC** 地址能够继续使用

- 堆叠线的连接

StackWise 技术采用特殊的堆叠线连接位于各交换机背面的堆叠端口，以此来实现智能堆叠功能。这里最关键的地方在于连接结构，堆叠线必须设计为环状结构才行。假设我们要构成三台交换机的堆叠，那么就应该设计成如下图所示的环状结构。

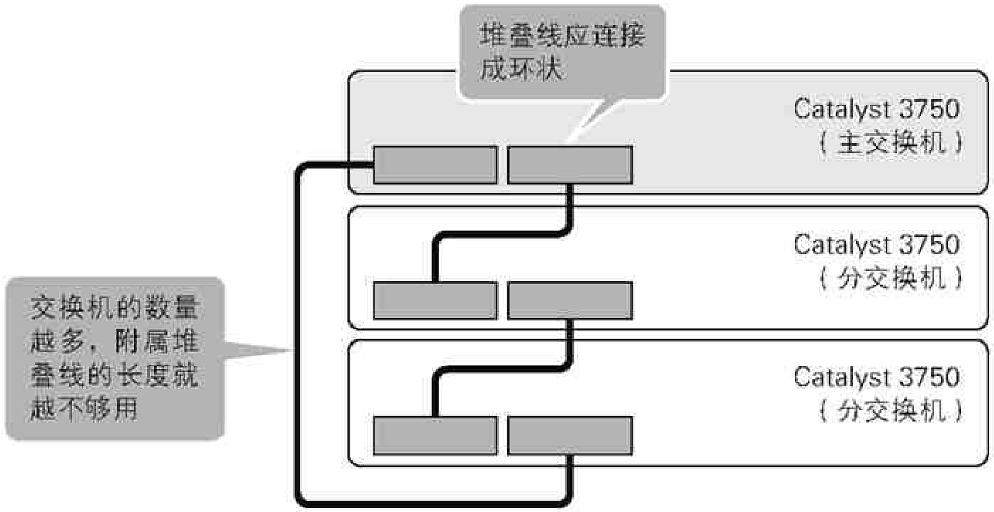


图 **4.1.22**　堆叠线应连接成环状

我们还需要在物理层面管理好堆叠线。交换机的数量一多，附属堆叠线的长度就会不够用。所以，如果交换机超过三台，最好事先就另买几条较长的堆叠线备用。另外，还得注意跨机架布线的情况，堆叠线因其自身的特性，并不适合跨机架布线，建议尽量在同一机架中完成布线作业。

[7] FIB（Forwarding Information Base，转发信息库）表是从路由表中提取出的表，其中仅包含了传输数据包所需的信息。

VSS

VSS 是一项可以用于 Catalyst6500 系列和 Catalyst4500-X 系列的冗余技术。它用多条 10G 链路连接两台 Catalyst6500，将它们集结成一台逻辑交换机。从物理层面来看有两台交换机存在，但是从网络的角度来看就像只有一台交换机在工作，只需进行一种设置，管理对象也只有一个。

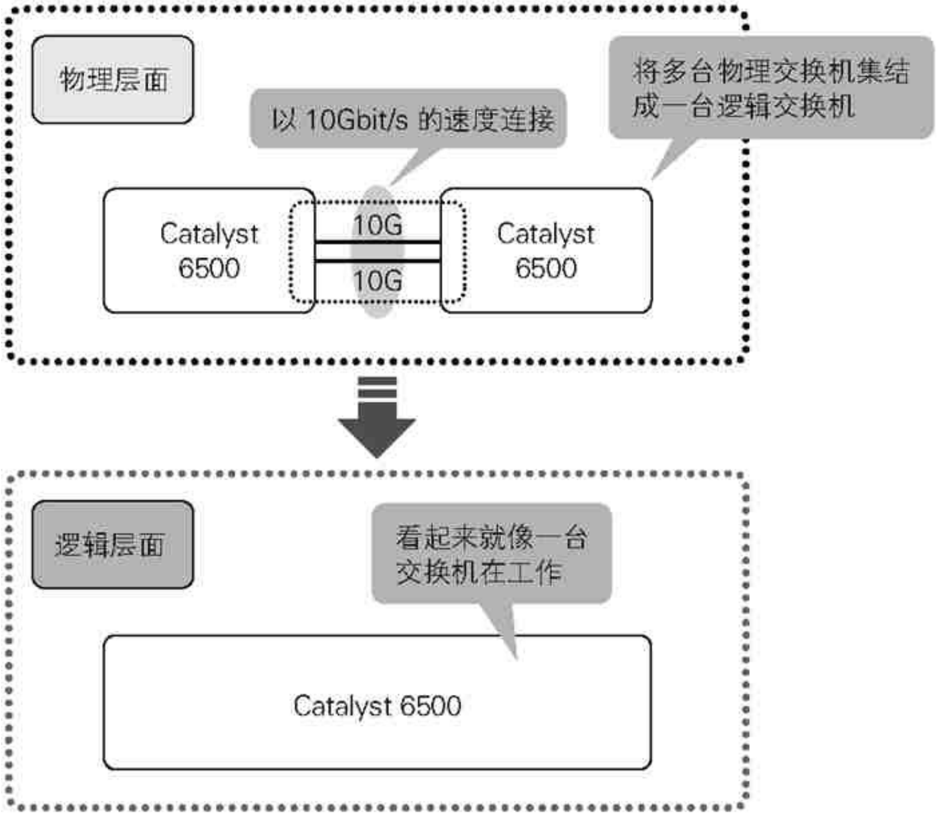


图 **4.1.23**　将两台物理交换机集结成一台逻辑交换机

- 虚拟域 **ID**

虚拟域 ID 是一个在逻辑层面上管理 VSS 对的 ID。在构建 VSS 的物理交换机之间设置相同的域 ID，就会生成 VSS 域。虚拟域 ID 用于 PAgP 或 LACP 的控制比特流，在网络内部必须设计成独一无二的才行。因此，连接不同 VSS 对的时候要多加注意，勿使域 ID 发生冲突。

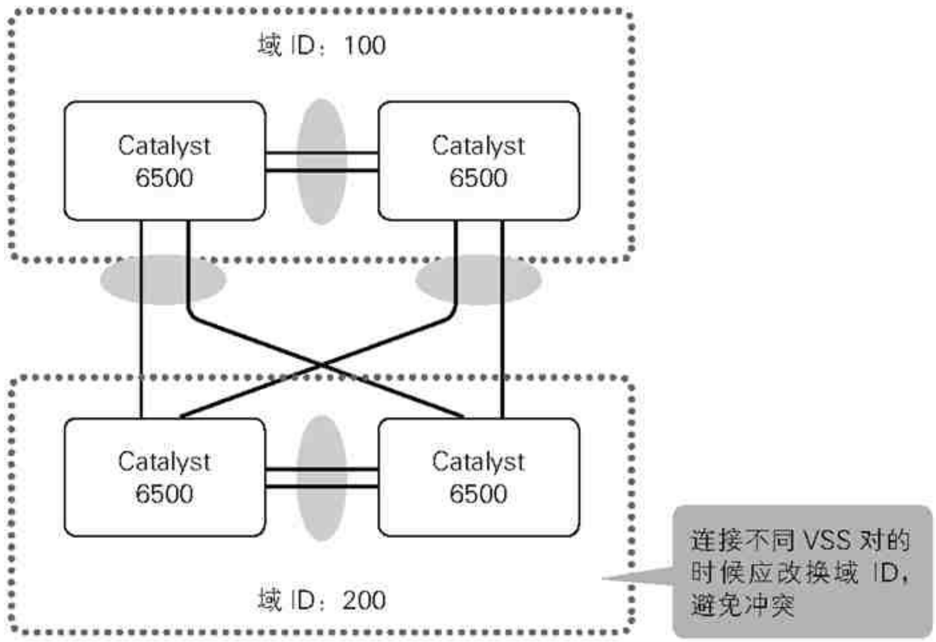


图 **4.1.24**　域 **ID** 在网络内部必须是独一无二的

- **VSL**

VSS 冗余技术的关键所在，就是连接两台交换机的 VSL（Virtual Switch Link，虚拟交换链路）。这个连接一般通过链路聚合将 10G 链路集结起来完成的。构建 VSS 需要交换特定的控制信息和同步信息，VSL 不仅肩负着交换这些信息的重任，在发生故障时，它还会成为数据的传输路径，其重要性不言而喻。

这部分的要点在于如何配置构成链路聚合的物理链路。我们应将一条链路用在管理引擎上，另一条链路用在其他线卡上，以此来应对线卡发生故障的情况。

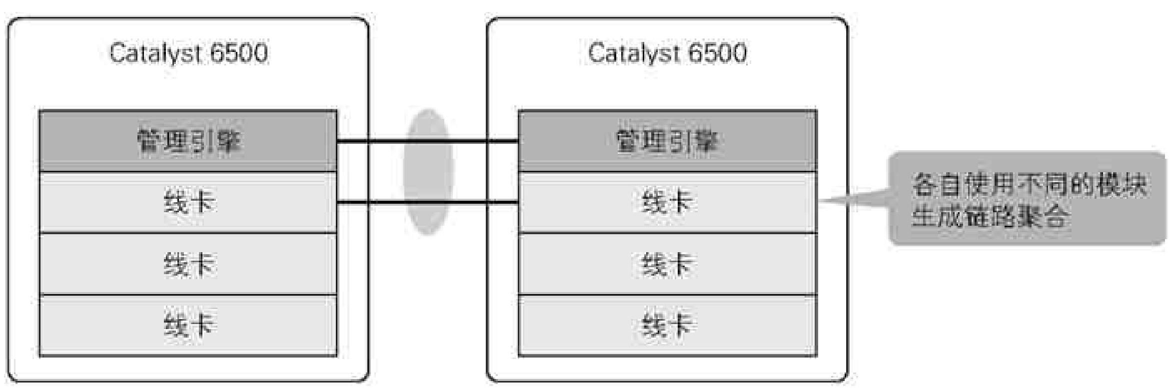


图 **4.1.25**　注意物理链路的配置

- 传输处理由活动交换机和备用交换机共同完成

VSS 身兼双职，根据不同的处理转换角色，既能实现冗余备份，又能增强传输能力。

路由协议控制和管理控制等由软件完成的、比较复杂的处理是由活动交换机和备用交换机共同完成的。一般情况下由活动交换机执行处理并将数据同步到备用交换机中去。活动交换机一旦发生故障，备用交换机就会升级为活动交换机，迅速完成接替，保证冗余效果。

而像传输比特流等在硬件上进行的简单处理则是由两台活动交换机完成的。两台物理交换机共同分担处理任务，以此来保证最大的传输能力。当其中一台物理交换机宕机时，另一台物理交换机就会立刻接手继续处理任务。

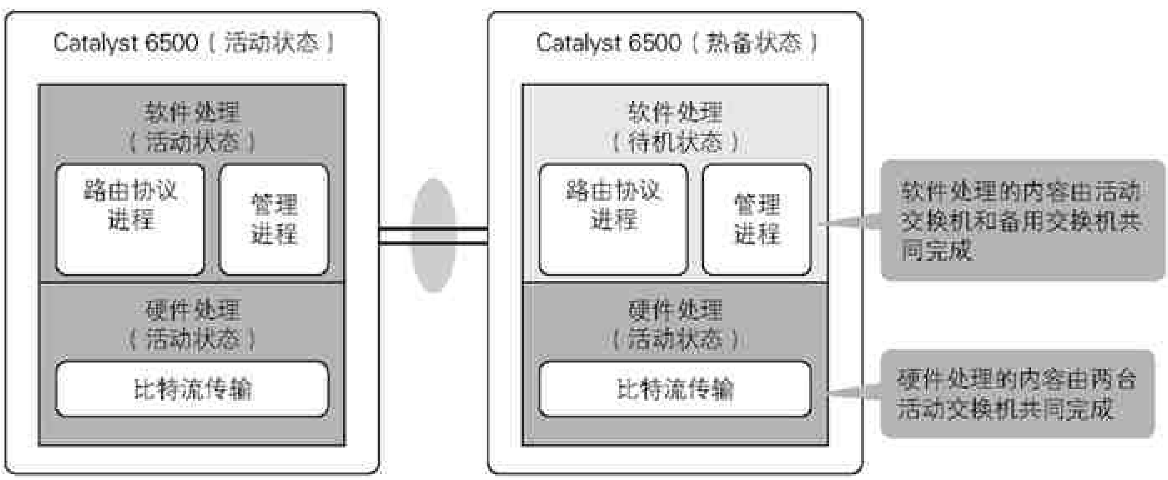


图 **4.1.26**　确保冗余备份和传输扩展

- **MAC** 地址

对任何冗余结构，我们都需要考虑到 MAC 地址这个细节。VSS 默认使用各台物理交换机自身的 MAC 地址，因此，假如我们重启两台物理交换机使活动交换机发生了切换和交接等状态的改变，那么，只要相邻设备的 ARP 切换尚未结束，通信就会一直处于中断状态，这可是足以致命的大问题。为了避免出现这样的局面，我们

在构建 VSS 的时候，应将默认设置改为使用虚拟 MAC 地址。这样，VSS 的 IP 地址和 MAC 地址就能保持不变，就不会受到来自相邻设备的 ARP 的任何影响了。总之，我们应尽量避免不必要的状态变更。

注意链路聚合的配置

将多台物理设备集结成一台逻辑设备这种类型的冗余技术大多是和链路聚合一起设计的。这时候有一点我们必须注意，那就是构成链路聚合的物理链路的配置问题。一定要从不同的物理交换机获取链路才行。以 VSS 结构为例，在同一台物理交换机上获取所有物理链路是毫无意义的，因为一旦该物理交换机宕机，通信就会断掉。因此，务必要从不同的物理交换机获取链路，以应对物理交换机发生故障的情况。

对于类似 Catalyst6500 系列的机箱式交换机，我们还必须注意同一台物理交换机内的物理链路配置问题。如果同一个线卡占用了所有的物理链路，那么一旦该线路卡出问题通信就会断掉。因此，我们务必让每一个线路卡各自取用一条物理链路，以备线路卡发生故障的不时之需。

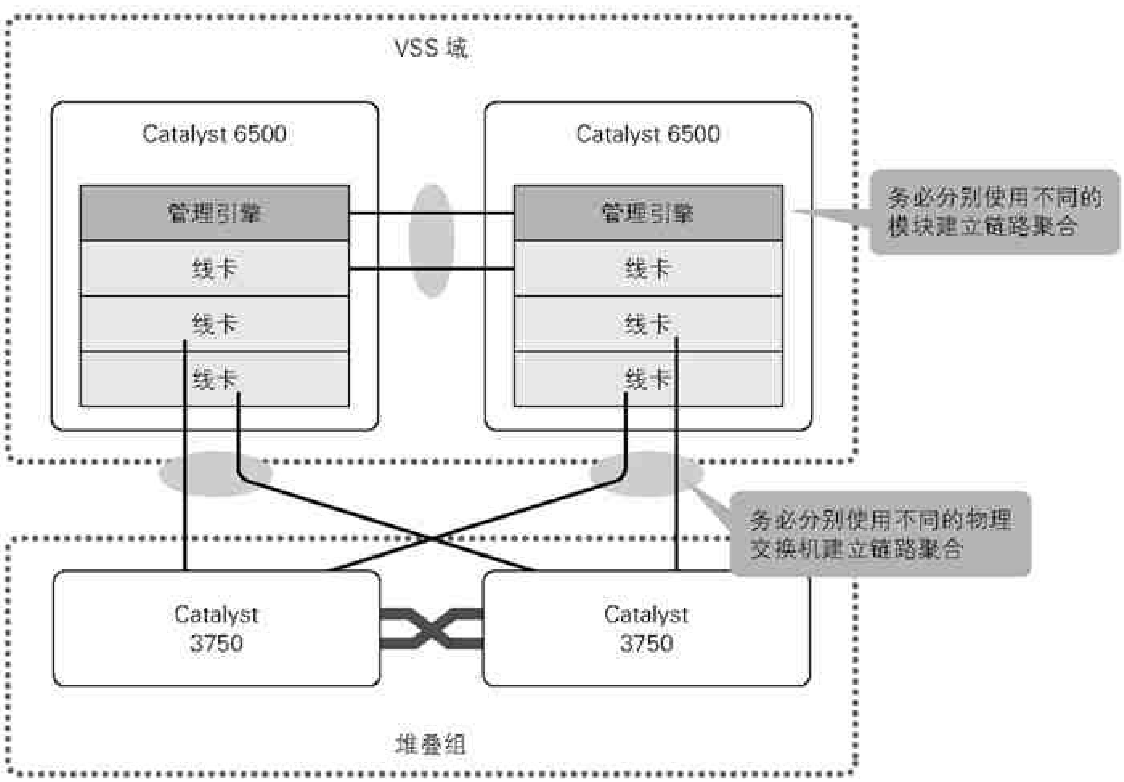


图 **4.1.27**　注意物理链路的配置问题

StackWise 技术和 **VSS** 的优点

StackWise 技术和 VSS 之所以会成为高可用性设计的主角，是因为它们除了冗余备份之外还具有别的优点。以往高可用性设计的主角是 STP（Spanning-Tree Protocol，生成树协议）[8]，然而 STP 在实现冗余备份的同时还不得不面临一些进退两难的窘境，因此长期以来一直令工程师们难于取舍。StackWise 技术和 VSS 则能够一举解决这些进退两难的问题，如今，它们早已成为了网络技术中不可或缺的存在。

> [8] 关于 STP 的内容将在 4.1.2 节中详细讲解。

本书将会介绍它们所具有的“增强传输能力”“简化结构”和“便于运行管理”这三个优点，并和使用 STP 时的情况分别进行比较，孰优孰劣，一目了然。

- 增强传输能力

  STP 是一种协议，它阻塞形成环回拓扑的某处端口，并按照逻辑树状结构去构造网络拓扑。由于它会阻塞某处端口导致数据无法传输，因而不能百分之百地发挥交换机原本具备的传输能力。与此相对，采用 StackWise 技术或 VSS 的网络结构中根本就没有阻塞端口存在，所有端口都能用到。

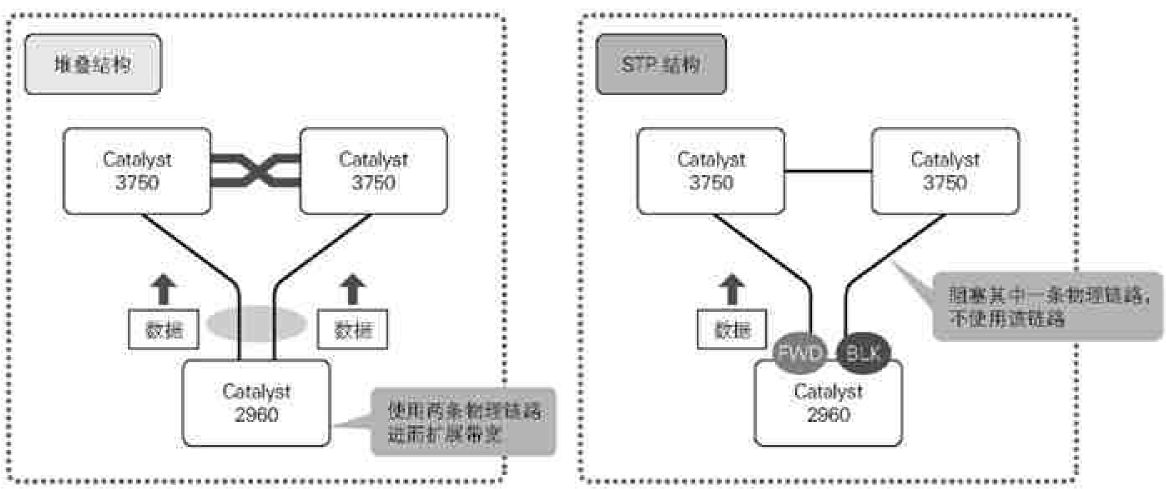


图 **4.1.28**　物理链路全部都能用上

- 简化结构

STP 对交换机的连接是呈环状的，随着网络规模的扩大，结构会变得越来越复杂。而且我们还得注意阻塞端口位于何处，这越发使得结构难于理清。与此相对，采用 StackWise 技术或 VSS 时，由于多台物理交换机集结成一台逻辑交换机，从逻辑上来说我们只需连接一台交换机即可，这能够大大地简化结构。

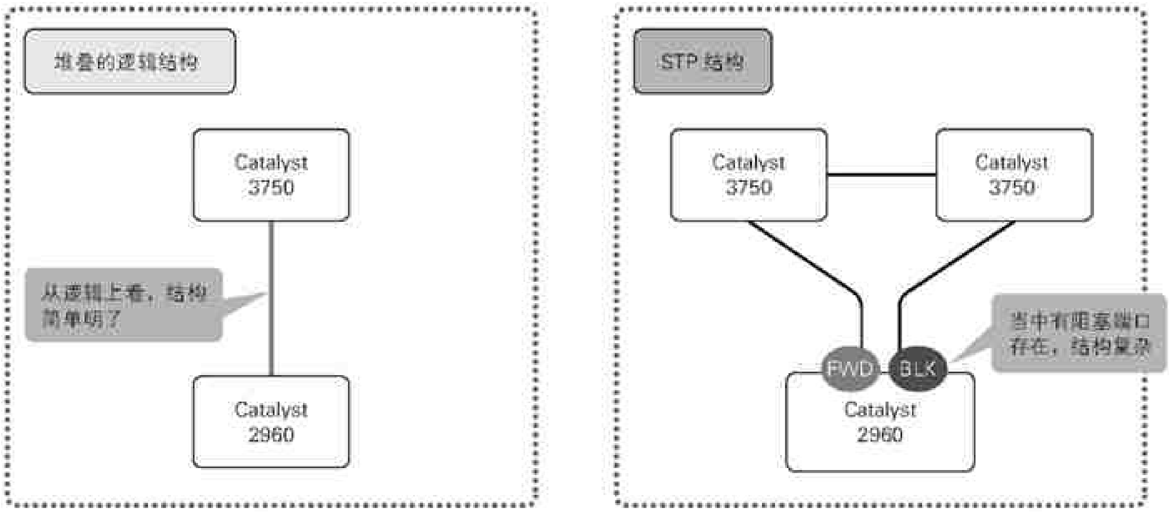


图 **4.1.29**　结构简单明了

- 便于运行管理

对网络管理人员来说，管理对象的增加仅仅意味着更加繁重的工作负荷。如果我们采用 STP，那么各台交换机都是独立工作的，有多少台交换机就会有多少个管理对象。例如，新增四台物理交换机就等于新增了四个管理对象。与此相对，采用 StackWise 技术或 VSS 时，我们只有堆叠组或 VSS 域这一个管理对象。即使新增四台物理交换机并将它们堆叠起来，从逻辑上说仍然只有一台交换机，所以只是新增一个管理对象而已。和采用 STP 的情况相比，运行管理要轻松很多。

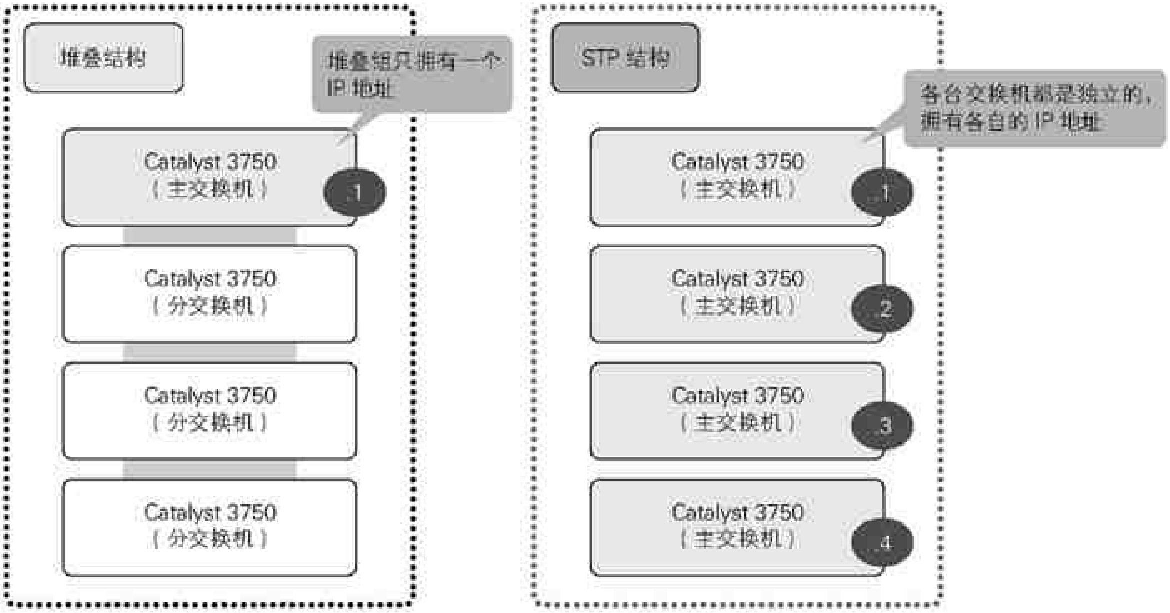


图 **4.1.30**　管理对象少

4.1.1.4　当上行链路中断时，让下行链路也随之中断

下面介绍一项性质比较特殊的物理层冗余技术，它叫作链路聚合故障转移，有些生产商称之为链路状态追踪或者 UFD（Uplink Failure Detection，上行链路故障检测），实际上都是同一个意思。链路聚合故障转移是指当上行链路（针对上层交换机的链路）中断时，让下行链路（针对服务器的链路）也随之中断。

网卡组合并不是万能的

正是因为大量的设备彼此融合、共同存在，网络的世界才得以成立。因此，我们设计时应始终参考相邻设备之间的连接结构。在仅靠网卡组合无法排除故障的特定环境中，链路聚合故障转移能够发挥巨大的作用。

我们来考虑一下图 4.1.31 所示的情况。这应该是一种非常常见的结构。在这种结构中，当交换机的上行链路中断时，通往上层的路径消失，导致通信无法进行。这里的网卡虽然绑定成了网卡组合的形式，但由于上行链路的中断和服务器并没有直接的关系，因而无法进行故障转移，服务器仍然会试图通过同一个物理网卡发送信息。

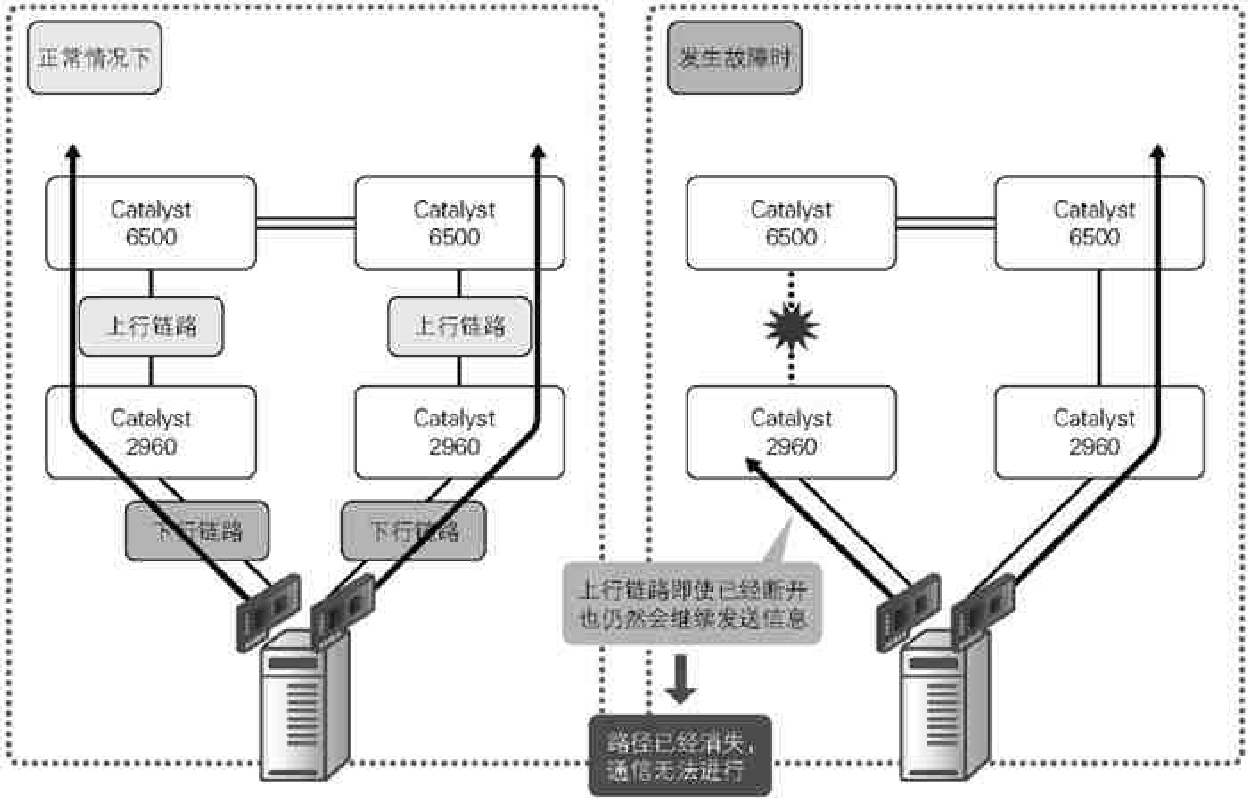


图 **4.1.31**　网卡组合无法检测到故障已发生

利用链路聚合的故障转移激发网卡组合的故障转移

想解决上述问题，就要利用链路聚合故障转移技术。链路聚合故障转移能够监控上行链路的物理链路状态并根据该状态控制下行链路。它的工作机制非常简单，那就是一旦发现上行链路中断，就让下行链路也随之中断。

那么，我们就来看看将链路聚合故障转移用在前面提到过的结构中会怎么样。上行链路一中断，链路聚合故障转移功能就会启动，强制性地关闭下行链路。于是整条链路中断，网卡组合启动故障转移机制，最终确保了通往上层的路径。

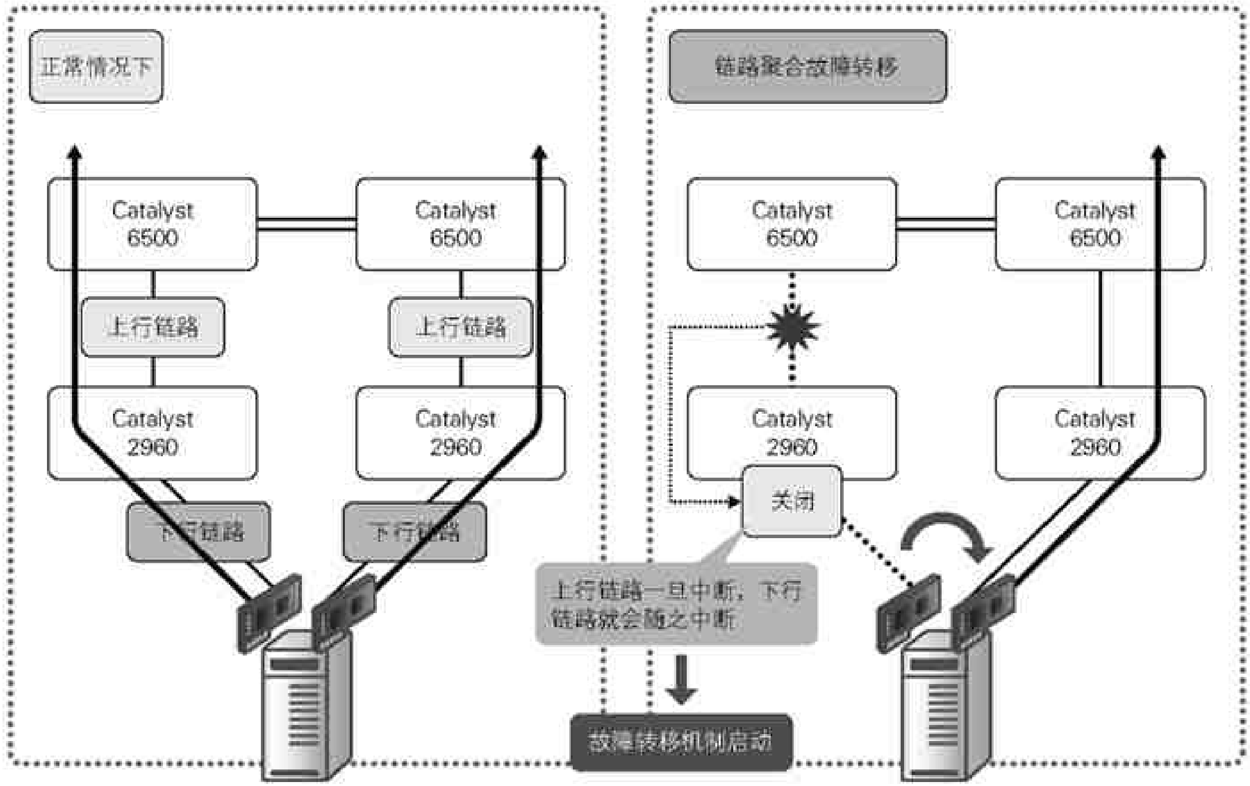


图 **4.1.32**　利用链路聚合故障转移强行确保通往上层的路径

4.1.2　数据链路层的冗余技术

关于数据链路层的冗余技术，我们只需掌握 STP（Spanning-Tree Protocol，生成树协议）就足够了。事实上，作为一项冗余技术，STP 早已是明日黄花、江河日下，不过它毕竟在过去的很长一段时期内都是网络高可用性的技术后盾，今后很可能还会继续存在下去，我们了解一下总归是有利无弊的。

4.1.2.1　**STP** 的关键在于根网桥和阻塞端口

STP 是一种协议，它阻塞形成环回拓扑的某处端口，并按照逻辑树状结构去构造网络拓扑，用于链路冗余和防止桥接组。这里我们要理解一个叫作 BPDU（Bridge Protocol Data Unit，网桥协议数据单元）的概念，BPDU 是相邻交换机之间传递的一种特殊管理帧，用于在网桥之间进行信息交换。STP 通过 BPDU 选择根网桥和阻塞端口，而 STP 的关键就在于根网桥和阻塞端口。

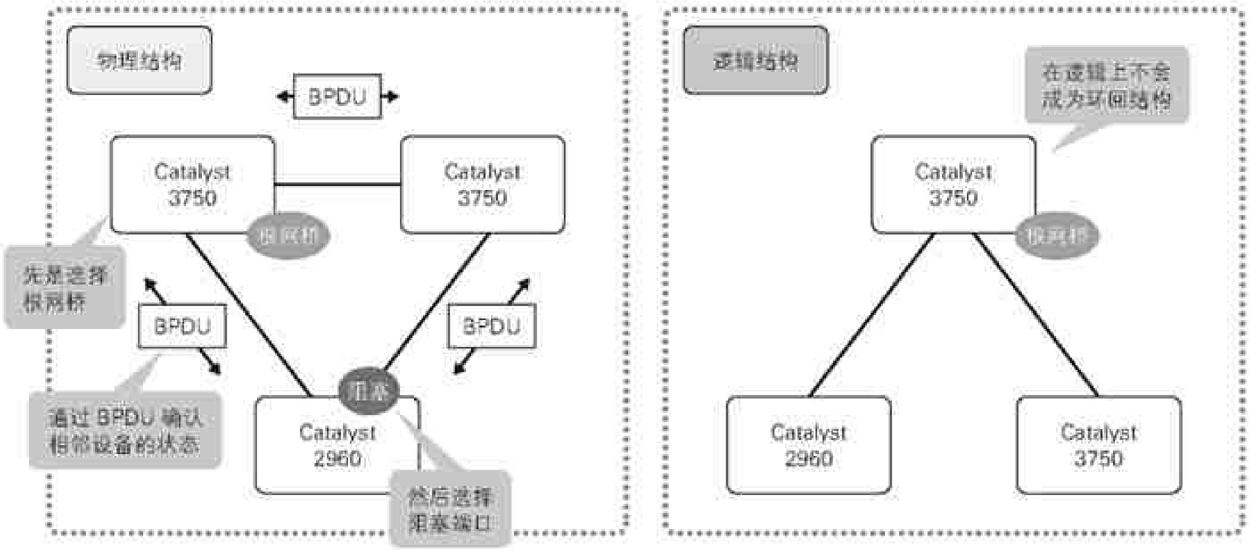


图 **4.1.33**　通过 **BPDU** 选择根网桥和阻塞端口

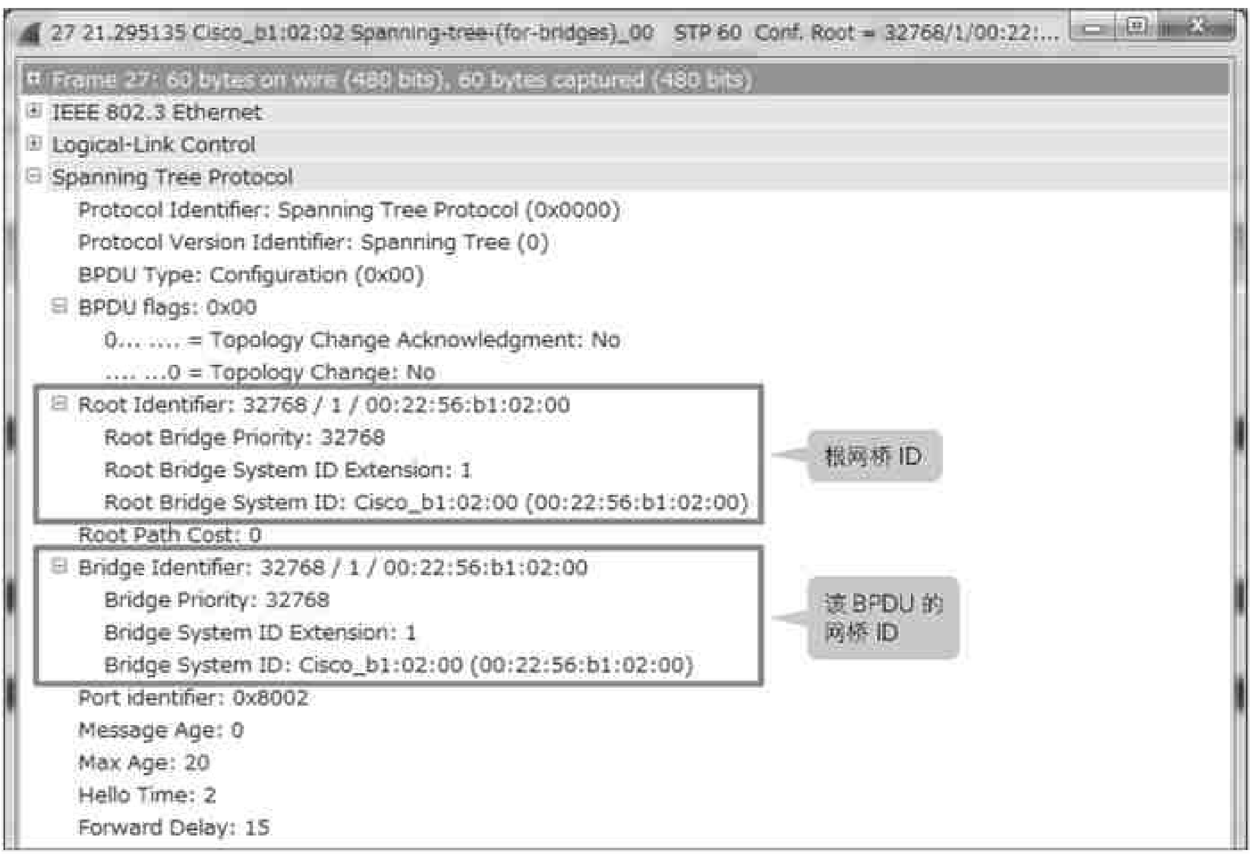


图 **4.1.34**　通过交换 **BPDU** 来选择根网桥和阻塞端口

根网桥取决于网桥优先级

根网桥是指由 STP 形成的、相当于逻辑树状结构根部（root）部分的交换机。离开了根网桥，STP 就无从谈起。根网桥是根据网桥 ID 选择出来的，而网桥 ID 则是由网桥优先级和 MAC 地址组成的。



图 **4.1.35**　根据网桥 **ID** 选出根网桥

STP 处于激活状态的交换机一经连接，就会交换 BPDU 并比较网桥 ID。比较并不是将网桥优先级和 MAC 地址作为一个整体进行的，而是有一定的先后顺序。首先是比较网桥优先级，网桥优先级最低的交换机将当选为根网桥，第二低的交换机则当选为第二根网桥。当根网桥发生故障时，第二根网桥就会顶替上去成为新的根网桥，因此第二根网桥相当于是一道为根网桥上的保险。如果网桥优先级都相同，那么就得比较 MAC 地址了，MAC 地址最小的交换机将当选为根网桥。

在真实的网络环境中，人们通常将网桥优先级设置成静态的，根据网桥优先级去选择根网桥，而不让 MAC 地址起决定性的作用。因此，对于根网桥和第二根网桥，我们应将它们的网桥优先级设置得比其他网桥低才行。

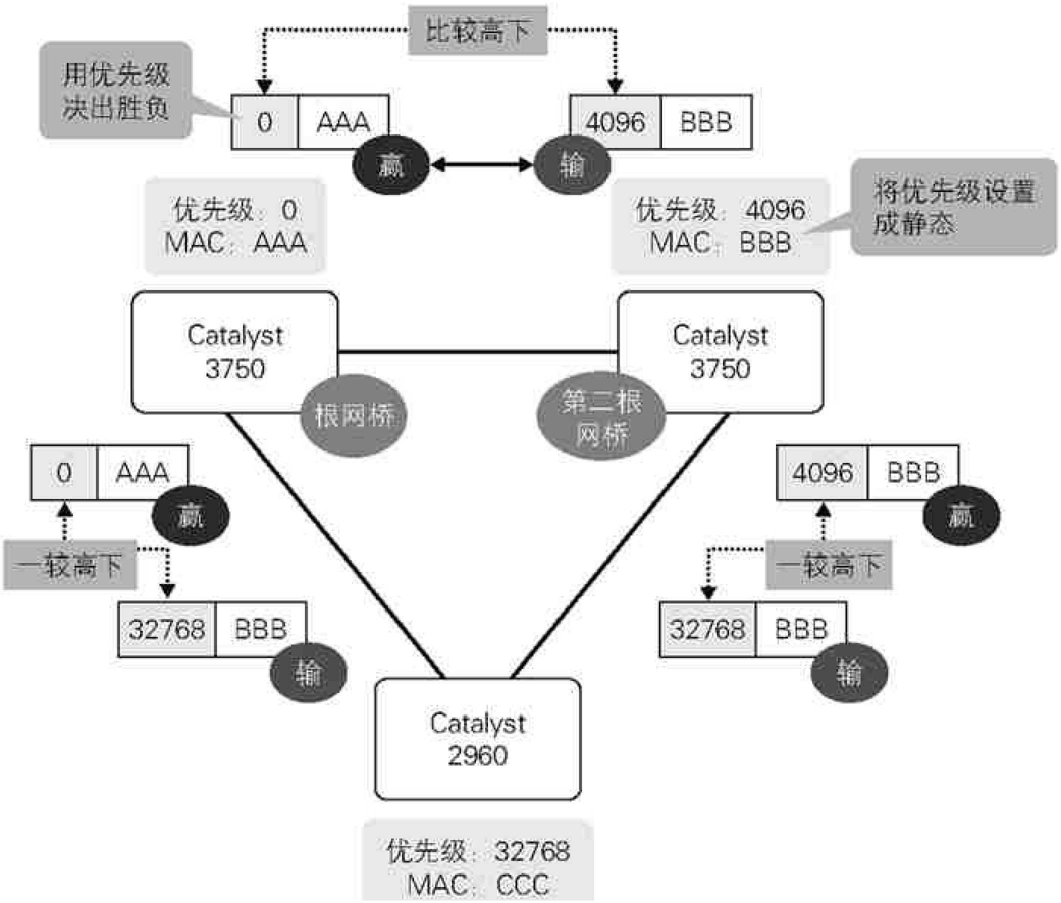


图 **4.1.36**　根网桥取决于网桥优先级

阻塞端口取决于路径开销

阻塞端口是端口的一种，其重要作用是让环回结构成为树状结构。和根网桥在逻辑上相距最远的端口就是阻塞端口。计算端口和根网桥之间的距离时要用到一个叫作路径开销的值，这个值是有具体规定的，不同的链路带宽有着不同的路径开销值，具体情况如下表所示。

表 **4.1.1**　路径开销的值取决于带宽

带宽	路径开销
10Mbit/s	100
100Mbit/s	19

1000Mbit/s（1Gbit/s）	4
10Gbit/s	2

我们根据 BPDU 中包含的路径开销算出各个端口到根网桥的距离，然后进行比较，数值最大的那个端口将成为阻塞端口。如果到根网桥的距离都相同，那就要比较网桥 ID 了，网桥 ID 最大的端口将成为阻塞端口。正常情况下阻塞端口并不传输数据，它是被完全排除在拓扑范围之外的。

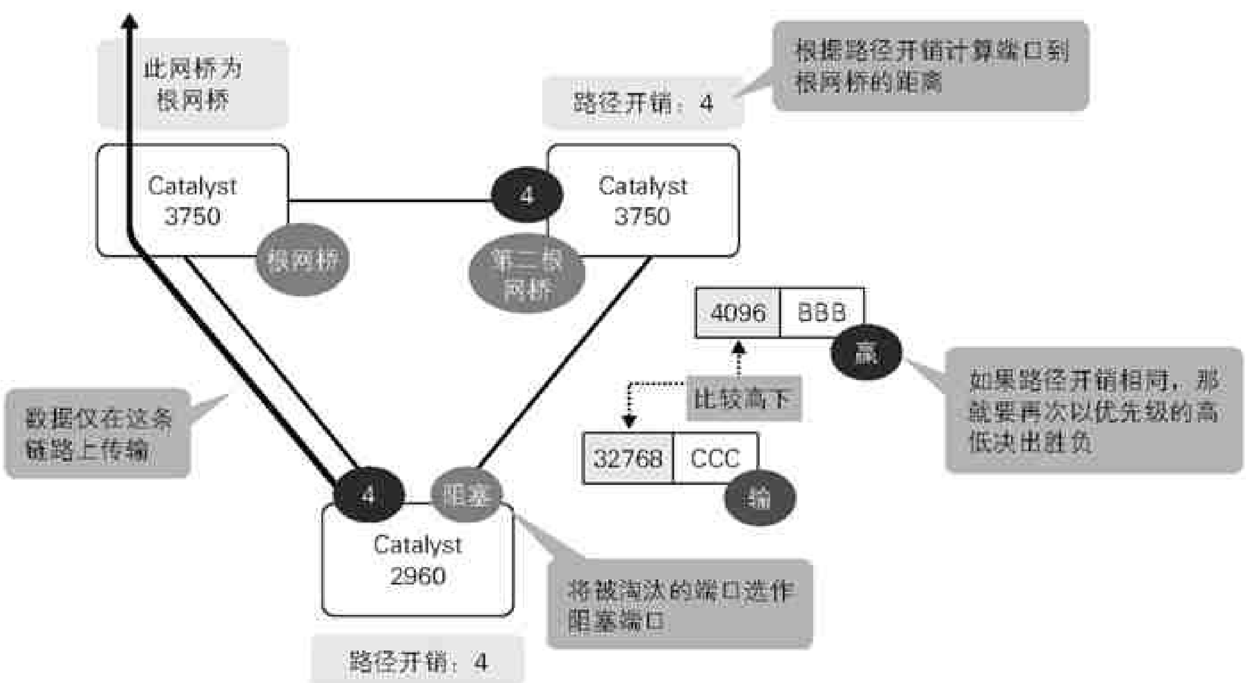


图 **4.1.37**　决定阻塞端口

- 固定逻辑链路的路径开销

  通过链路聚合连接交换机的时候，我们应注意逻辑链路的路径开销。按照默认的设置，构成逻辑链路的物理链路一旦中断路径开销就会改变，于是会重新进行 STP 计算，而这个重新计算的结果可能会导致阻塞端口发生变化。

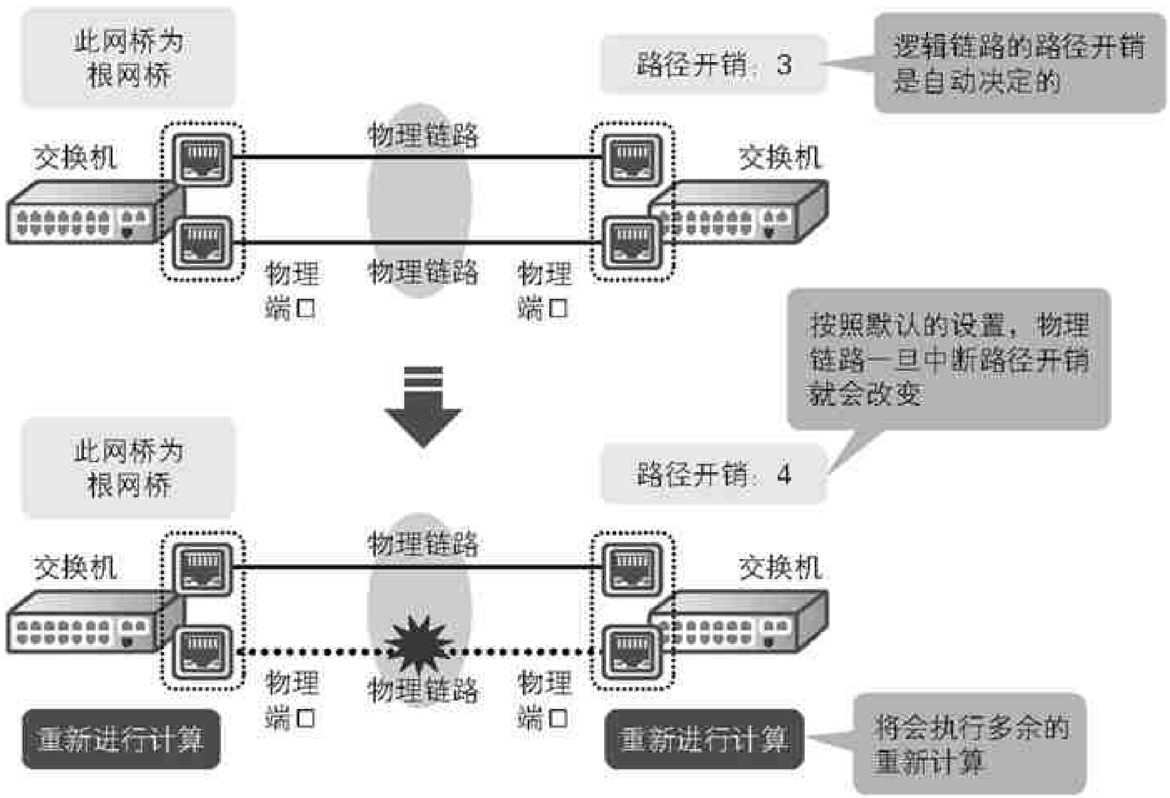


图 **4.1.38**　按照默认的设置，物理链路一旦中断路径开销就会改变

为了避免多余的重新计算，逻辑链路的路径开销对逻辑链路应设置成静态的才行。比方说，如果使用的是 Catalyst 交换机，通过设置链路聚合会产生一个叫作 Port-Channel 接口的逻辑端口，我们应该在该逻辑端口中设置路径开销。这虽然只是一个细节问题，但却非常重要。

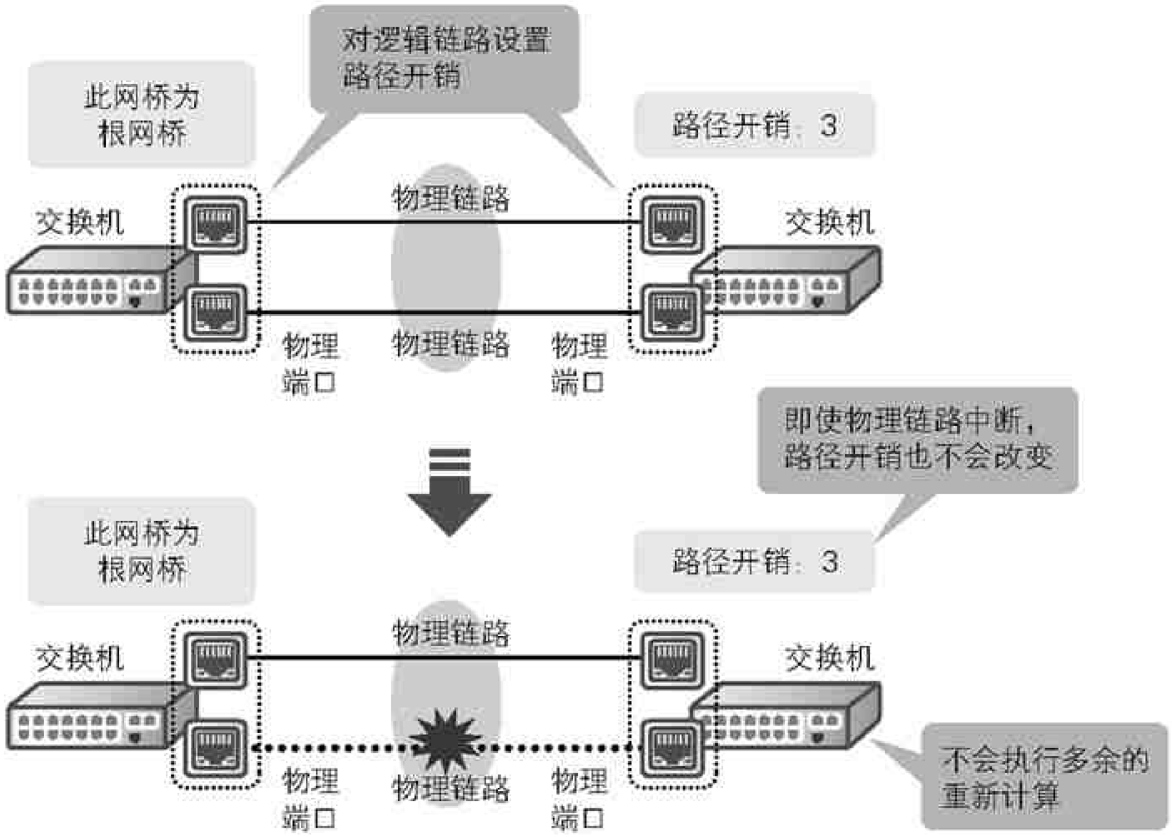


图 **4.1.39**　对逻辑链路设置路径开销

以 **VLAN** 为单位进行负载均衡

BPDU 是以 VLAN 为单位生成的管理帧，因此，根网桥和阻塞端口也是以 VLAN 为单位决定的。STP 在通信的处理和带宽的负载均衡中很好地利用了这种设计。

- 通信处理的负载均衡

  在拥有众多 VLAN 的环境中，如果我们把所有的根网桥都汇集到一台交换机上，是绝对无法进行高效处理的。根网桥掌握着 STP 的一切，可以说是首领一样的交换机。把所有 VLAN 中的 STP 处理都交给一台根网桥交换机的话，处理负荷会过于集中。所以，我们务必为每个 VLAN 都选择不同的根网桥，尽量让负载得到均衡的分配。

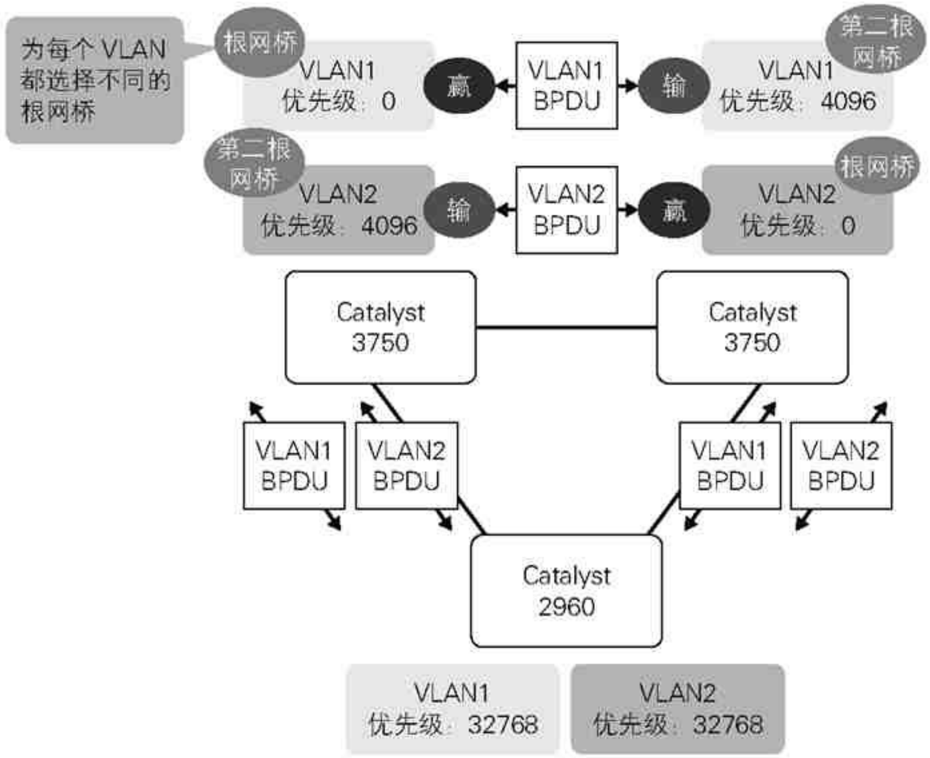


图 **4.1.40**　为每个 **VLAN** 都选择不同的根网桥，让负载得到均衡的分配

- 带宽的负载均衡

前面已经讲过，根网桥也决定着阻塞端口。因此，如果将所有的根网桥都集中到一台交换机上，那么所有 VLAN 的阻塞端口就会落到同一处，这会使通信流量过于集中在某条特定的链路上。因此，我们必须为每个 VLAN 都选择不同的根网桥和阻塞端口。改变根网桥能让 VLAN 的通信流量走另外的链路，最终达到带宽负载均衡分配的效果。

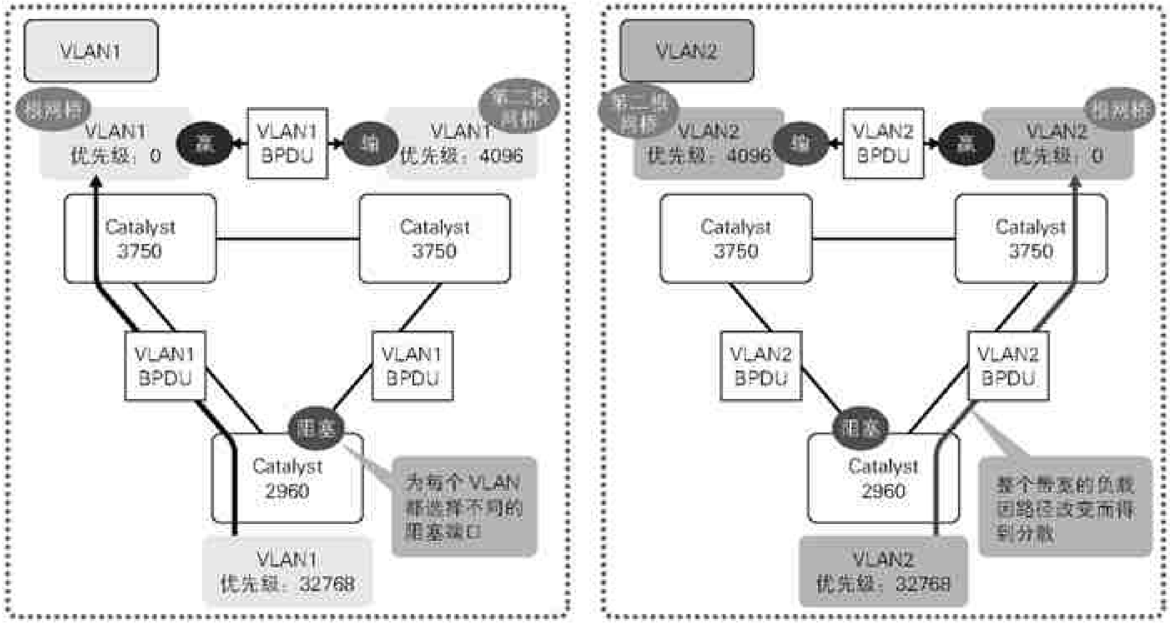


图 **4.1.41**　为每个 **VLAN** 都选择不同的路径，让带宽负载得到均衡的分配

应将阻塞端口安排到这里！

通过 STP 实现链路冗余时，其标准的物理结构以及相应的设计基本上是约定俗成的。标准的物理结构有三角形结构和四方形结构两种，下面就详细讲解在这两种结构中，分别需要将阻塞端口安排到何处。

- 三角形结构

  三角形结构是由根网桥、第二根网桥和非根网桥构成的，呈三角状，是最容易理解也最为常见的一种结构。在这种结构中，非根网桥连接第二根网桥的端口会被阻塞。

  在三角形结构中，所有交换机的根路径开销（到根网桥的路径开销合计值）都是一样的。因此，只要通过网桥优先级选出根网桥和第二根网桥，我们的目标端口就会被自动阻塞。这里的关键在于根网桥和第二根网桥的位置。

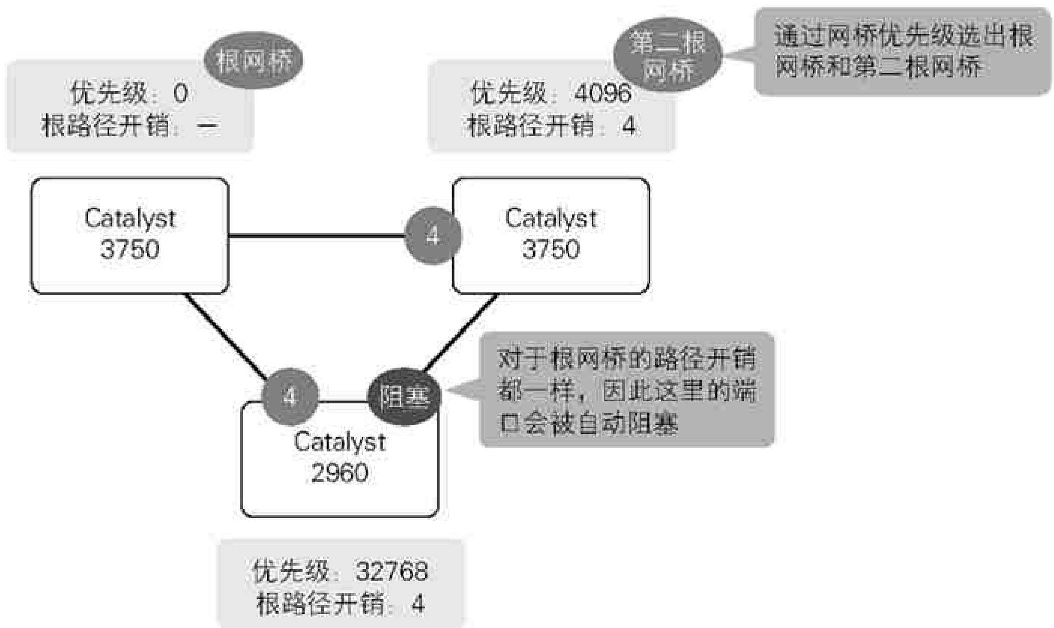


图 **4.1.42**　在三角形结构中只需决定根网桥和第二根网桥即可

- 四方形结构

  四方形结构是由根网桥、第二根网桥和两台非根网桥构成的，呈四方形，极为少见。在这种结构中，连接不同非根网桥的设备之间的通信并不经过上层，因此，被阻塞的是位于根网桥对角处的非根网桥中连接第二根网桥的端口。

  在四方形结构中，根网桥和位于根网桥对角处的交换机之间的路径有两条，它们的根路径开销是一样的。为了让连接第二根网桥的端口成为阻塞端口，我们必须对路径开销进行操作。具体方法如下：在经过第二根网桥的路径某处加上路径开销，或者是在经过并排非根网桥的路径某处减去路径开销。在四方形结构中，我们不仅要注意根网桥和第二根网桥的位置，还必须考虑路径开销。

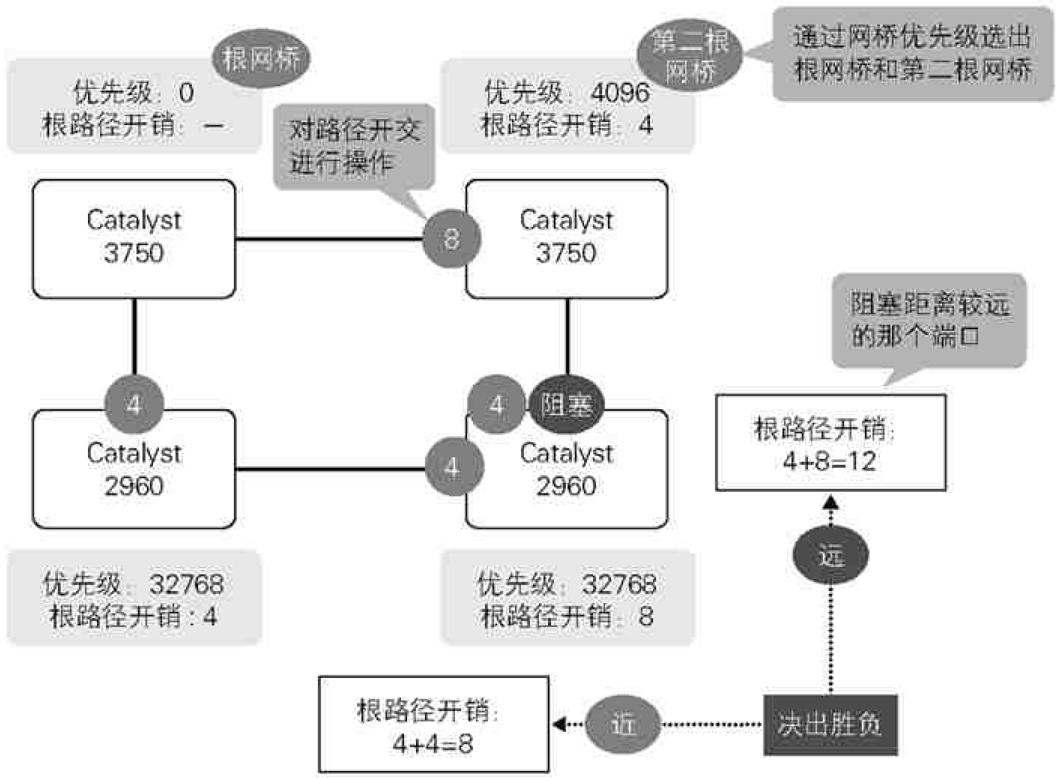


图 **4.1.43**　在四方形结构中要对路径开销进行操作

4.1.2.2　STP 有三种

STP 有三种，分别是 PVST（Per VLAN Spanning-Tree，每 VLAN 生成树）、RSTP（Rapid Spanning-Tree，快速生成树协议）和 MST（Multiple Spanning-Tree，多生成树）。其中，PVST 是最造出现的，后来为了填补 PVST 的缺陷又相继出现了 RSTP 和 MST。它们的工作机制各不相同，但设计重点都在于收敛时间。收敛时间是指端口状态从切换之后到稳定下来所需的时间。

表 **4.1.2**　**STP** 分 **PVST**、**RSTP** 和 **MST** 三种

STP 的种类	**PVST**	**RSTP**	**MST**
规格	思科公司独有	IEEE802.1w	IEEE802.1s
收敛时间	长	短	短

收敛方式	基于定时器	基于事件	基于事件
BPDU 的单位	VLAN	VLAN	MST 区域
根网桥的单位	VLAN	VLAN	实例
阻塞端口的单位	VLAN	VLAN	实例
负载均衡的单位	VLAN	VLAN	实例

人们一般使用 **PVST**

PVST 是 STP 中最常用的一种，其基本原理在前面已经介绍过了。各交换机先是交换 BPDU 并选出根网桥，然后再选出阻塞端口，等这些处理都稳定下来之后就通过互相发送 BPDU 定期监控彼此的状态，如果有一方在一定时间之内没有收到 BPDU，或是收到了来自根网桥的、表示拓扑变更的 BPDU（TCN BPDU），系统就会认为发生了故障，并执行几道重新计算之后将阻塞端口释放。

作为一种协议，PVST 曾经是初期 STP 环境的坚强后盾。然而它有一个足以致命的缺点，那就是收敛时间太长。PVST 基于“Hello 定时器”（2 秒）、“最大失效定时器”（20 秒）和“转发延迟定时器”（15 秒）这三个定时器执行收敛处理，它们都会直接影响收敛时间的长短。

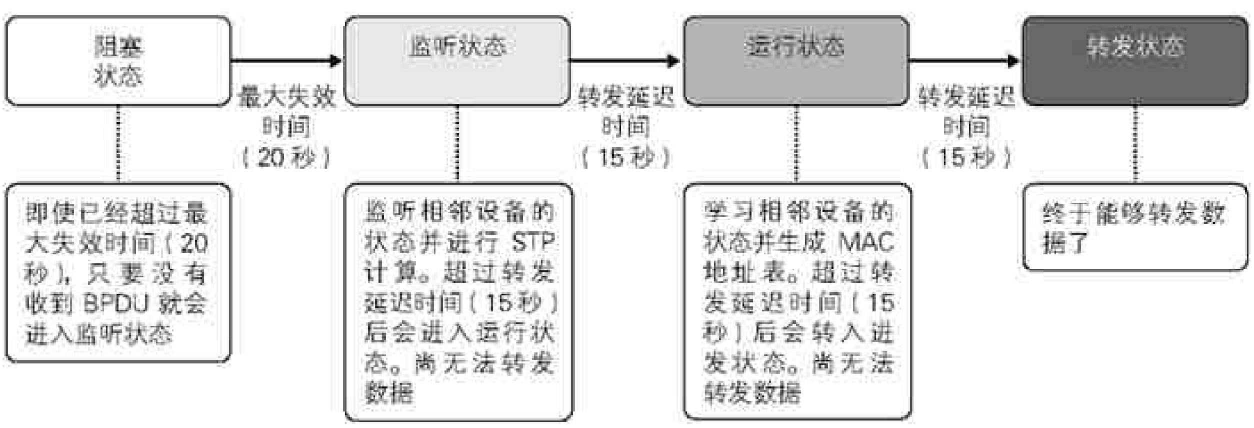


图 **4.1.44**　**PVST** 基于定时器执行收敛处理

如果处理必须按照一定的顺序执行，前一个处理结束之后才能进入下一个处理，那么收敛时间肯定是会比较长的。PVST 的所有端口都是按照阻塞、监听、运行、转发这个顺序进行计算的，到释放阻塞端口为止，需要 50 秒钟的时间（20 秒＋ 15 秒 ×2），而 50 秒对于关键任务环境来说太长了，RSTP 和 MST 就是为了弥补这个缺点而出现的。

RSTP 能够快速完成收敛工作

RSTP 是为了弥补 PVST 收敛时间较长这个缺点而出现的，它由 IEEE802.1w 定义。RSTP 通过发送 BPDU 执行两个分别叫作提案和同意的握手处理，由此迅速掌握对方的状态。

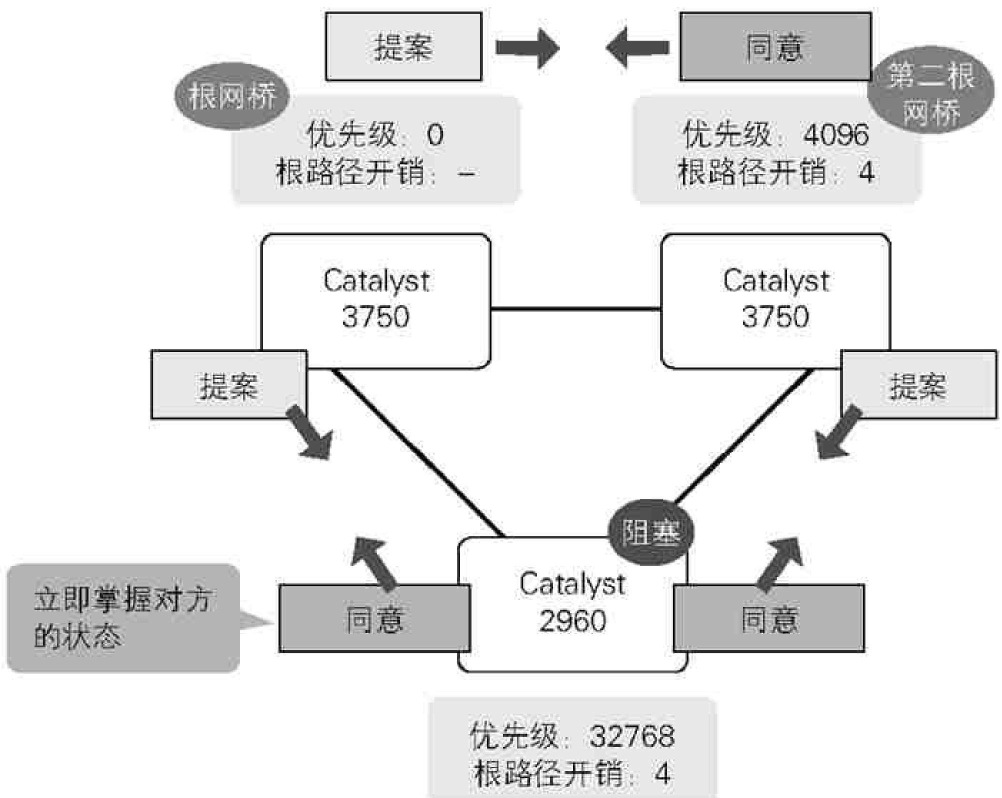


图 **4.1.45**　通过提案和同意处理来掌握对方的状态

经过改良，RSTP 处理故障的速度也比 PVST 提高了很多。如果是拥有阻塞端口的交换机发生了直接性的链路故障，RSTP 会立即释放阻塞端口。如果是没有阻塞端口的交换机发生了链路故障，则由发生故障的交

换机对经过 TC（Topology Change，拓扑变更）置位的 BPDU 进行泛洪处理，向整个网络通知拓扑结构的变化。接下来，通过提案和同意再次执行握手处理，然后立即释放阻塞端口。RSTP 不像 PVST 那样基于定好的定时器顺序去逐一处理，所以不必等前一个的处理结束之后才开始下一个。它是受事件触发而启动的，仅需短短一秒钟的时间便能完成收敛工作。除非是极度敏感的应用程序，否则都不会觉察觉到有过这样一瞬而过的通信中断。

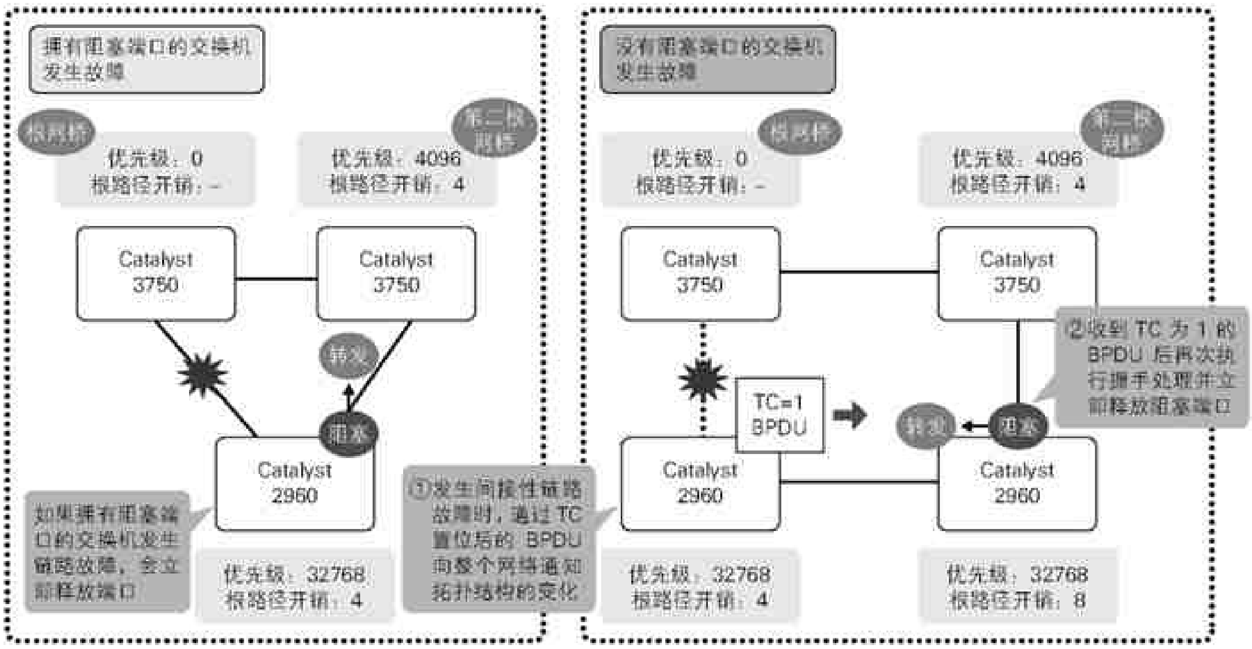


图 **4.1.46　RSTP** 能够迅速进行切换

MST 能够同时兼顾快速和高效

MST 和 RSTP 一样，也是为了弥补 PVST 收敛时间较长这个缺点而出现的一种协议。它由 IEEE802.1s 定义，基本机制也和 RSTP 一样，通过提案和同意去掌握对方的状态，受事件触发而启动，目的是缩短收敛时间。不过相比 RSTP，MST 还多出了一个新的概念，那就是用来提高处理效率的“实例”。粗略地讲，实例就是由多个 VLAN 构成的群组。PVST 和 RSTP 都是以 VLAN 为单位运作的协议，因此，当环境中有大量 VLAN 存在时我们就需要一个个地去处理和管理。而 MST 是在每个 MST 区域 [9] 中生成 BPDU 并以实例为单位运作，这样，我们就能以实例为单位去执行处理和管理了，不光处理本身，整个运行管理的效率都能获得提高。

[9] MST 区域是指拥有相同的 MST 设置且彼此连接的交换机的群集。

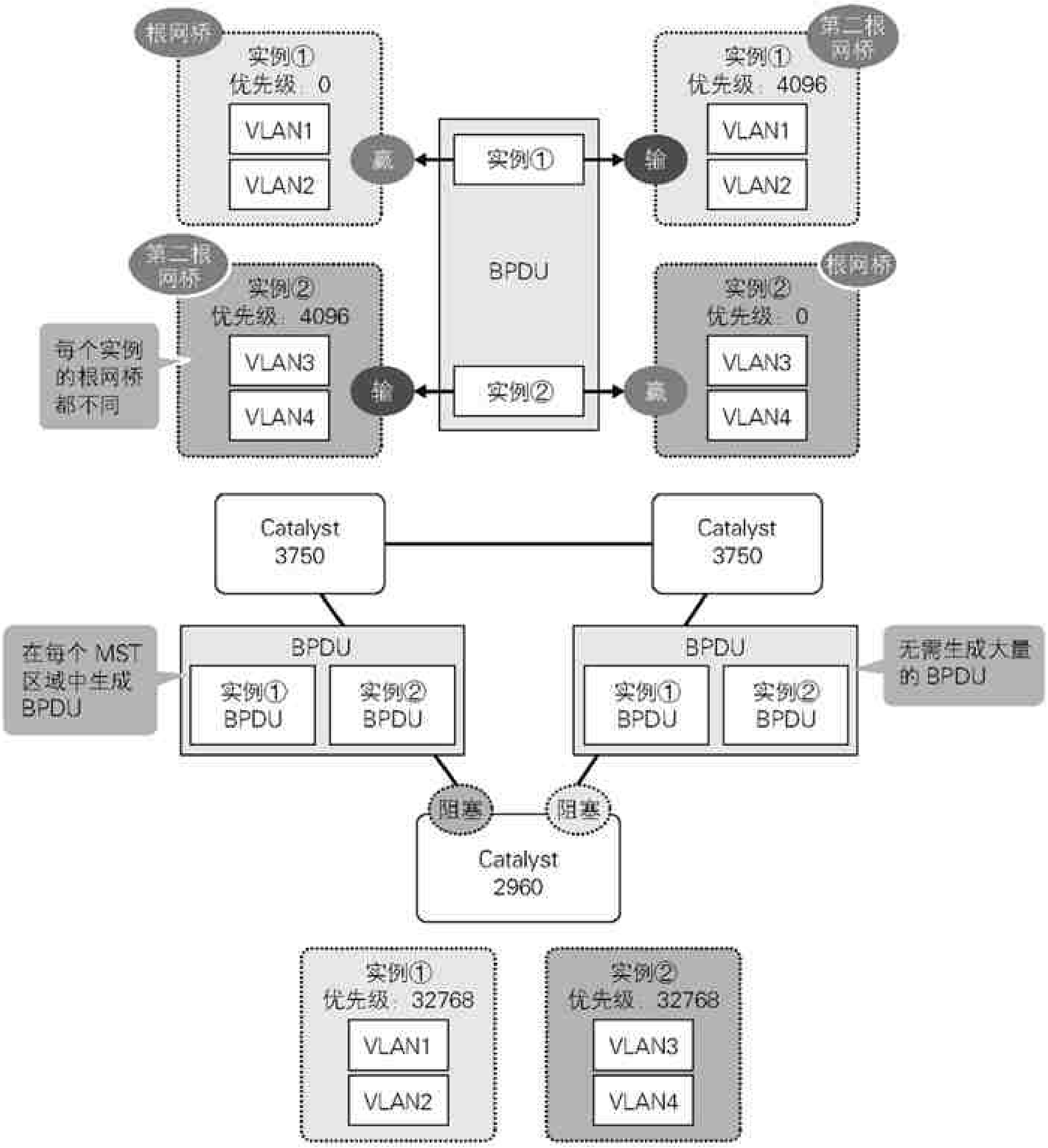


图 **4.1.47**　**MST** 以实例为单位进行处理

4.1.2.3　同时启用多项可选功能

STP 并不是单枪匹马发挥作用的，有很多备选功能可供使用，我们应根据实际的网络环境选择合适的可选功能。下面就介绍一下最近在网络环境中非常常见的 PortFast 和 BPDU 守护这两种可选功能。

PortFast 功能

将 STP 激活之后，所有的端口都会按照阻塞、监听、运行、转发这个顺序去进行计算。这些计算结束之后比特流才能被转发出去，大约需要 50 秒钟的等待时间。然而，对于连接服务器和 PC 的端口这种发生环回的可能性非常低的端口来说，这些计算是毫无必要的。于是人们为 STP 开发出了 PortFast 功能，它能让端口连接成功之后马上就进入转发状态。只要我们激活 PortFast 功能，端口就能迅速进入转发状态，因此连接一成功就可以马上转发比特流了。

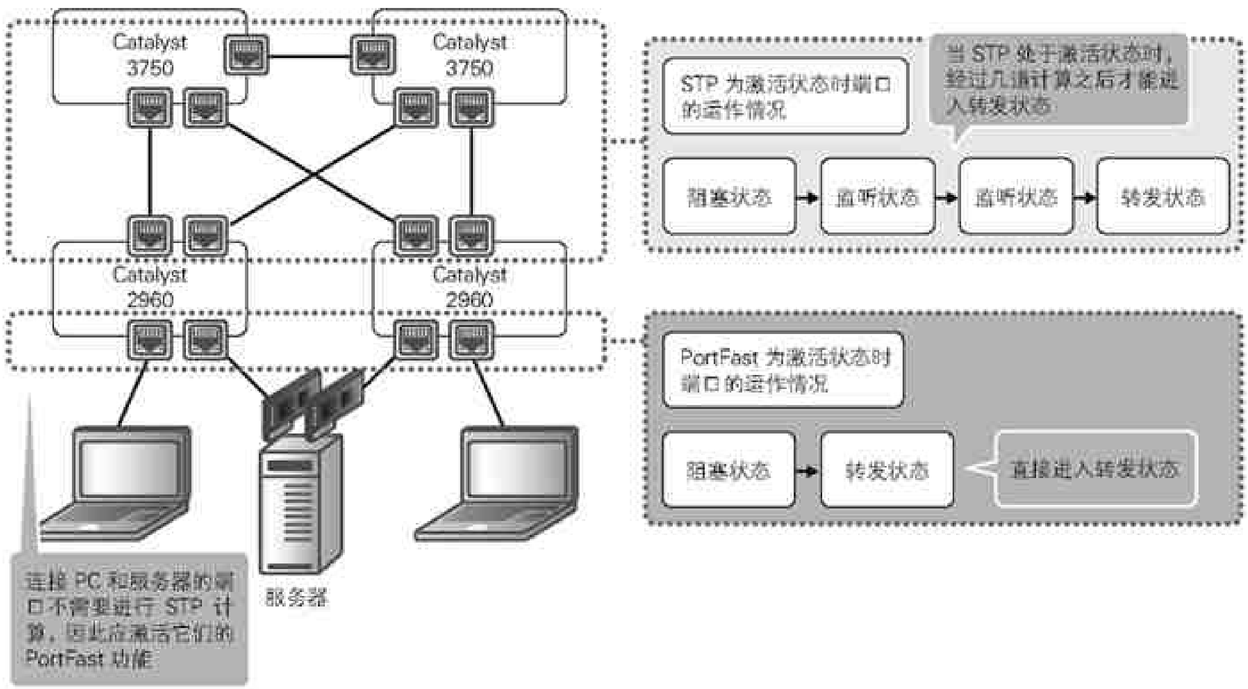


图 **4.1.48**　对于连接终端和服务器的端口应激活它们的 **PortFast** 功能

有些服务器的使用手册中可能写有“请勿激活 STP 功能”这样的字句，但却没有给出详细的解释。明明是需要通过 STP 去实现高可用性的环境，却又不允许激活 STP 功能，这是不可能的。如果看到这样的使用手册，我们可以认为这是在要求我们在连接的端口中设置 PortFast 功能。

BPDU 守护功能

BPDU 守护是指设定了 PortFast 功能的端口一旦收到 BPDU 就会强制关闭该端口的功能。前面已经提到过，PortFast 功能是针对那些不太可能发生环回的、连接 PC 和服务器的端口而设置的。然而，事实上我们谁也无法保证环回一定不会发生。例如，我们在运行管理中偶尔会听到这

样的事：用户将原本用于连接 PC 的端口擅自接到了集线器上，结果导致了环回的发生。为了避免出现这种情况，我们应开启 BPDU 守护功能。

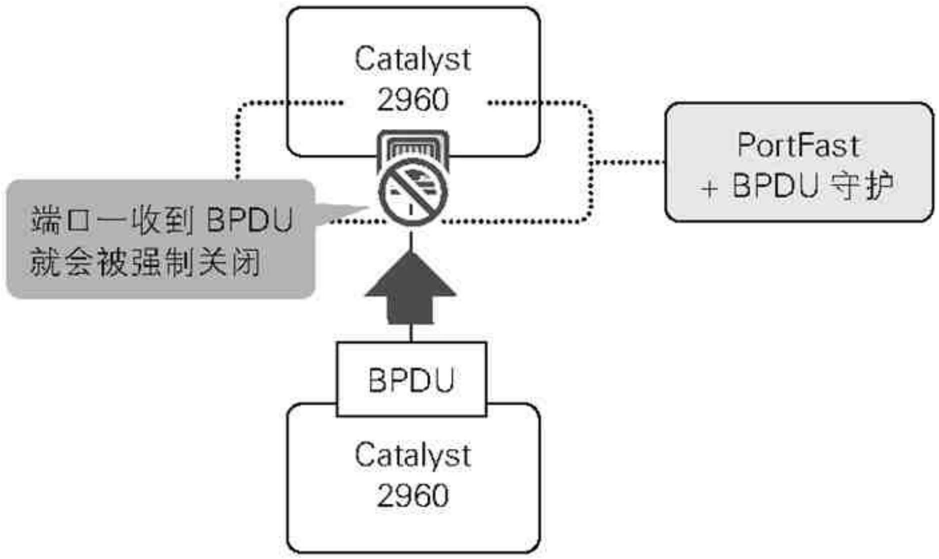


图 **4.1.49**　端口收到 **BPDU** 时就会被强制关闭

4.1.2.4　利用 **BPDU** 切断桥接环路

如今，STP 在冗余技术中早已风光不再，但这并不意味着它即将销声匿迹，因为它有一个看家本领，那就是能够避免桥接环路的产生。今后，STP 很可能会凭借这一特色继续存在下去。

桥接环路是足以致命的

首先，我们来看看什么是桥接环路。桥接环路是一种以太网帧在路径上环回不止的现象，它是由物理层面和逻辑层面的环回结构造成的。交换机会对广播进行泛洪处理，于是，当有环路存在的时候，广播被转发出去之后遇到泛洪处理，就会被无休止地重复转发，最终导致通信中断。

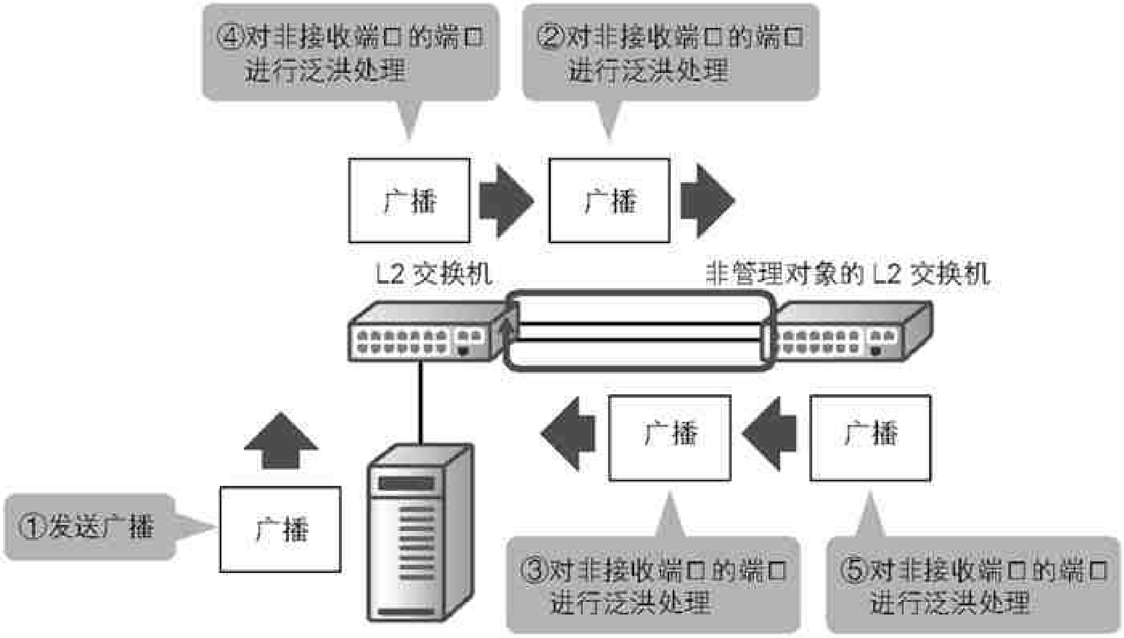


图 **4.1.50**　广播环回不止

以前，笔者负责的某位客户曾经捅过这样一个娄子——他对一个来路不明的集线器做了环回连接，结果导致桥接环路的产生，出了事故。用他的描述就是“集线器机房打雷了”，这句话给我留下了深刻的印象，简直就是我心目中不可磨灭的名言。想来“打雷”应该是所有交换机端口的 LED 灯一起闪光了的缘故，那还真是“电闪雷鸣”的场面啊。当时，由于整栋楼的核心交换机起着核心路由选择的作用，整栋楼的 7000 个端口同时断掉，通信在眨眼之间就全面陷入了瘫痪，实在是太可怕了……所以，我们绝不能让桥接环路发生。尤其是近年，考虑到运行管理的便利性，人们往往会采用“利用核心路由选择将 VLAN 尽量做大”的设计理念，然而在这样的环境中一旦发生桥接环路，其影响之大、后果之严重将不堪设想。对此，我们一定要备有万全之策。

利用 **BPDU** 守护功能切断环路

我们可以利用 BPDU 守护功能来切断桥接环路。桥接环路是很少发生在网络管理人员的管理范围内的，而是大多发生在服务器和用户端口这些管理人员管理范围之外的地方，的确从人的天性来说，看到闲置的端口就会想要派上用场吧。

对于连接服务器和用户终端的端口，我们应激活它们的 PortFast 功能，同时设置 BPDU 守护功能。BPDU 守护功能的机制是，当激活了

PortFast 功能的端口收到 BPDU 时会被强制关闭。如果有环路存在，计划之外的 BPDU 就会被发送到 PortFast 功能已被激活的端口中去，BPDU 守护功能会马上截住这些 BPDU 并关闭端口，这样，环路就被切断了。

前面已经讲过，作为一项冗余技术，STP 早已被 StackWise 技术和 VSS 取代，不会再有大的作为了，然而它能够避免桥接环路这个特色还是会继续发扬下去的。

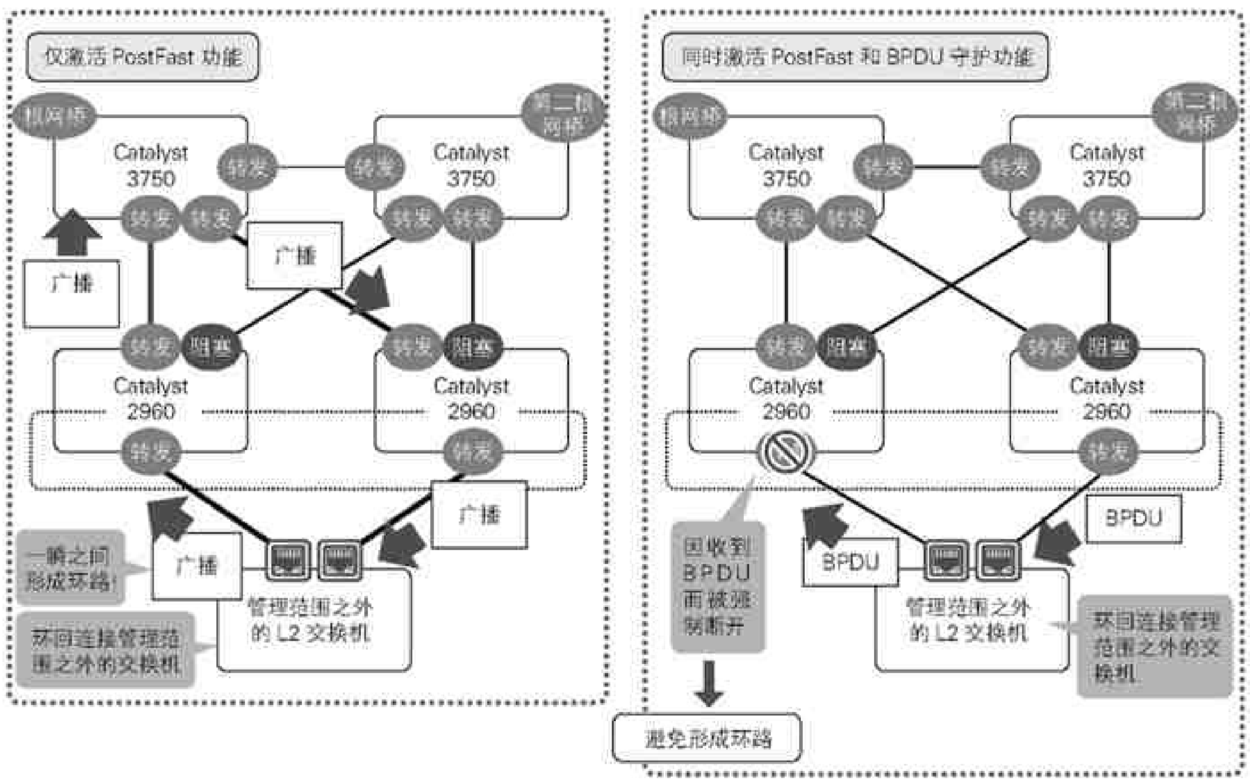


图 **4.1.51**　利用 **BPDU** 守护功能切断环路

不过，BPDU 守护功能并不是万能的。假设存在一台收到 BPDU 后将其丢弃的交换机，而有人又擅自环回连接上了该交换机，那么还是会导致通信中断，在一瞬之间形成环路的。因此，我们应该指定好管理规定，不用的端口务必要关闭掉。

4.1.3　网络层的冗余技术

关于网络层的冗余技术，我们只需掌握 FHRP（First Hop Redundancy Protocol，第一跳冗余协议）和路由协议就足够了。这两个协议在服务

器端互相合作，共同实现冗余备份。下面就分别讲解一下这两个协议。

4.1.3.1　FHRP

FHRP 是一种用于实现服务器和 PC 的第一跳，也就是默认网关冗余备份的协议。它在很早之前就被广为使用了，可以说是冗余备份的基础要素。目前，LAN 中的 FHRP 虽然正在被 StackWise 技术和 VSS 逐渐取代，但依然占有一席之地。此外，FHRP 在防火墙和负载均衡器的基础冗余技术中也有着用武之地，好好掌握它是有利无弊的。

两种常用的 **FHRP**

FHRP 使多个默认网关就像一个虚拟的默认网关那样运作，以此来实现冗余备份。它将两台设备共享的 IP 地址设为虚拟 IP 地址，然后将该虚拟 IP 地址当作默认网关使用。

可用于路由器和 L3 交换机的 FHRP 共有三种，分别是 HSRP（Hot Standby Routing Protocol，热备份路由协议）、VRRP（Virtual Router Redundancy Protocol，虚拟路由器冗余协议）和 GLBP（Gateway Load Balancing Protocol，网关负载均衡协议）。在这三种协议中人们一般会选用 HSRP 或 VRRP，下面本书就详细介绍这两种协议。

表 **4.1.3**　关于 **FHRP** 只需掌握 **HSRP** 和 **VRRP** 即可

FHRP的种类	**HSRP**	**VRRP**
组名	HSRP组	VRRP组
最大组数	255	255
构成组的路由器	活动路由器 备用路由器	主路由器 备份路由器
用于Hello数据包的多播地址	224.0.0.2	224.0.0.18

Hello间隔（发送Hello数据包的间隔）	3秒	1秒
Hold时间（判断出故障的时间）	10秒	3秒
虚拟IP地址	设成不同于真实IP地址的IP地址	可设成和真实IP地址相同的IP地址
虚拟MAC地址	00-00-0C-07-AC-XX（XX为组ID）	00-00-5E-00-01-XX（XX为虚拟路由器的ID）
Preempt功能（自动恢复）	默认为休眠状态	默认为激活状态
认证	可以	可以

- **HSRP**

  HSRP 是思科公司独有的 FHRP，如果网络环境采用的全是清一色的思科设备，那么考虑使用 HSRP 是绝对没错的。

  HSRP 通过 Hello 数据包中的组 ID 相互识别，然后比较优先级（优先的程度），将优先级最高的那台路由器设置为活动路由器，其他的设置为备用路由器。对于希望其成为活动路由器的设备，我们应将它的优先级设置得比备用路由器的更高。

  Hello 数据包实际上就是一种 UDP 多播。在源 IP 地址中写入各台路由器的真实 IP 地址，在目的 IP 地址中写入 224.0.0.2，在源 / 目的端口号中写入 UDP/1985，然后对构成 HSRP 所必需的这些信息进行封包处理。

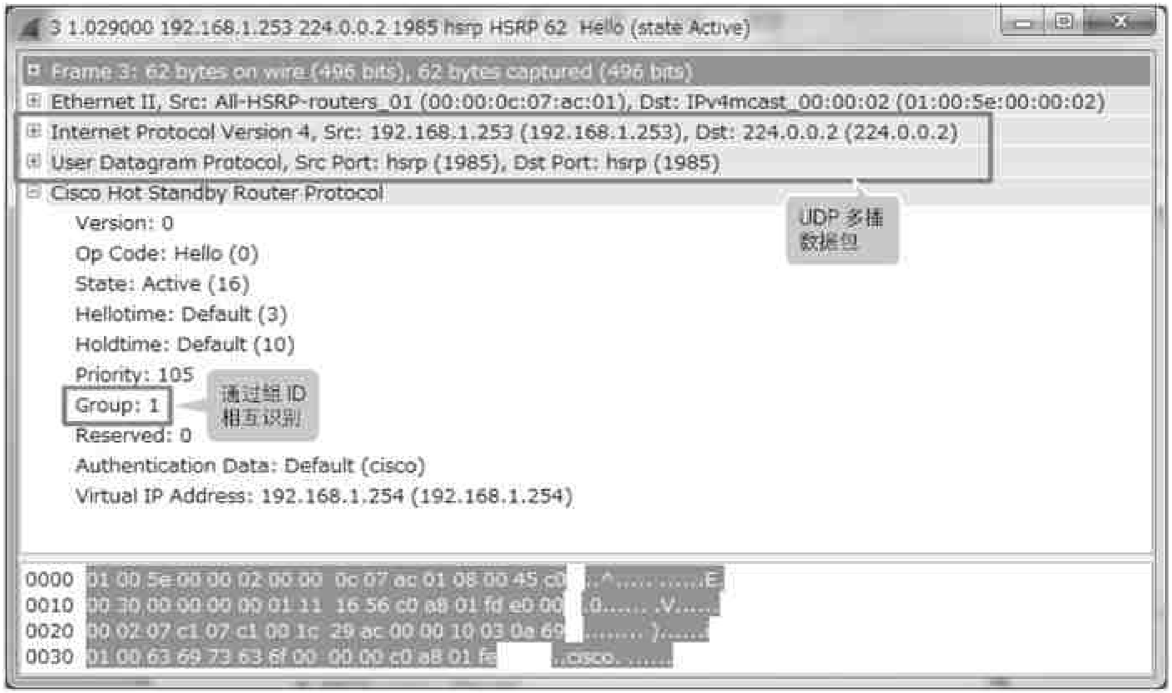


图 **4.1.52**　通过 **Hello** 数据包的组 **ID** 相互识别

一般来说，只有活动路由器才会接收用户的通信流量并基于路由表的信息去转发数据包。双方每隔 3 秒会互相发送 Hello 数据包，如果其中一方超过 10 秒未能收到 Hello 数据包，或是收到了优先级较低的 Hello 数据包，那么备用路由器就会接替原来的活动路由器工作，原理就是这么简单。

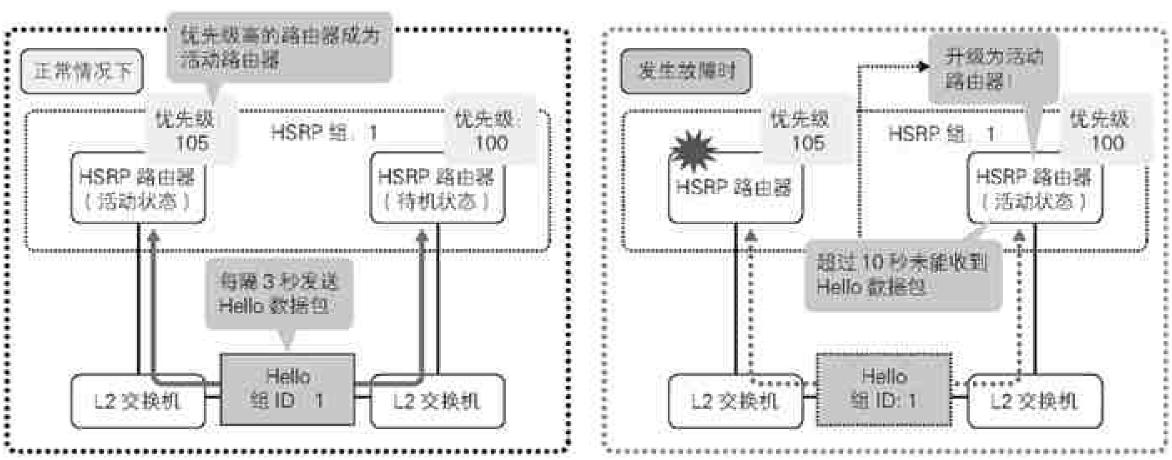


图 **4.1.53**　通过 **Hello** 数据包了解对方的状态

- **VRRP**

VRRP 是由 RFC 定义的 FHRP，一般用在由多家供应商提供的不同品牌设备所组成的环境中。我们可以认为 VRRP 的工作原理和 HSRP 的基本上是一样的，二者只是在名称和设计上有一些微妙的区别而已，我们只要弄清楚这些区别所在就可以了。

VRRP 通过 Advertisement 数据包中的虚拟路由器 ID 相互识别，然后比较优先级，将优先级最高的那台路由器设置为主路由器，其他的设置为备份路由器。对于希望其成为主路由器的设备，我们应将它的优先级设置得比备份路由器的更高。

Advertisement 数据包实际上就是一种多播。在源 IP 地址中写入各台路由器的真实 IP 地址，在目的 IP 地址中写入 224.0.0.18，然后对构成 VRRP 所必需的这些信息进行封包处理。



图 **4.1.54**　通过 **Advertisement** 数据包的虚拟路由器 **ID** 相互识别

一般来说，只有主路由器才会接收用户的通信流量并基于路由表的信息转发数据包。双方每隔 1 秒会互相发送 Advertisement 数据包，如果其中一方超过 3 秒未能收到 Advertisement 数据包，或是收到了优先级较低的 Advertisement 数据包，备份路由器就会接替原来的主路由器工作，原理和 HSRP 一样非常简单。

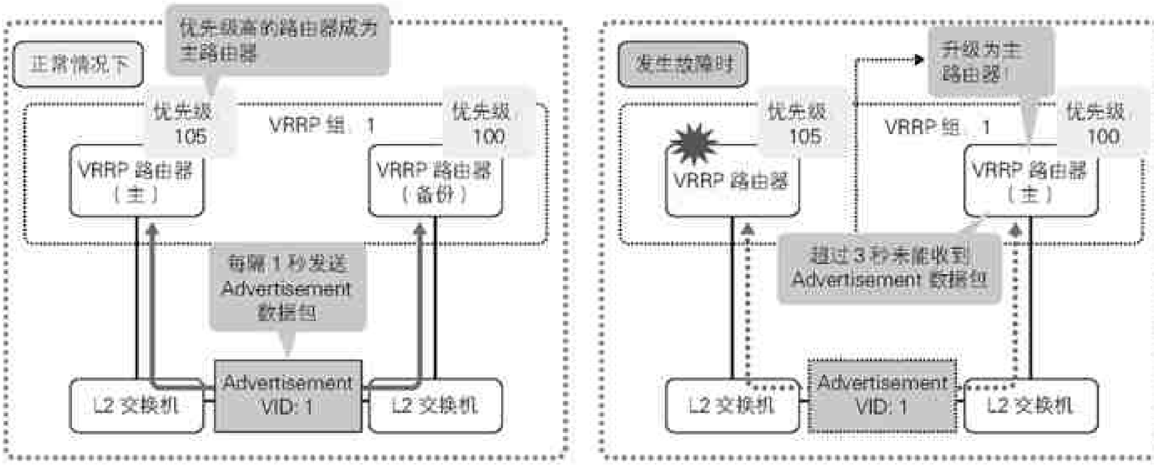


图 **4.1.55**　通过 **Advertisement** 数据包了解对方的状态

基于 **Hello** 报文的接收情况执行故障转移

当备用路由器在 Hold 时间内未能收到 Hello 报文，或是收到了优先级较低的 Hello 报文时，FHRP 就会执行故障转移。下面我们来分别了解一下这两种情况。

- **Hold** 时间超时

  备用路由器收不到 Hello 报文的情况也许比较容易想象。当活动路由器发生故障，或是发送 Hello 报文的接口有问题的时候，Hello 报文就会发不出去。于是备用路由器收不到来自活动路由器的 Hello 报文，FHRP 执行故障转移。

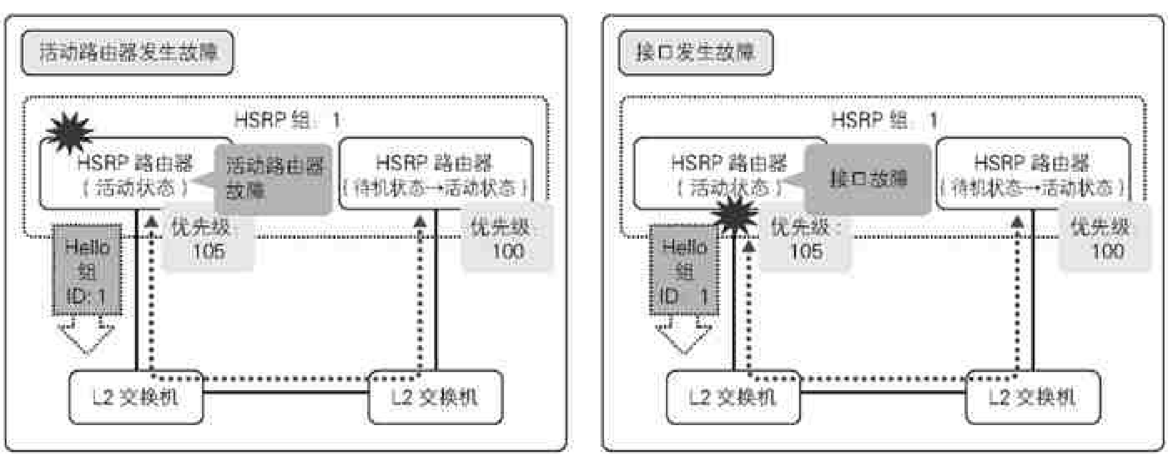


图 **4.1.56**　活动路由器或接口的故障导致 **Hold** 时间超时

- 收到优先级较低的 **Hello** 报文

  活动路由器和备用路由器是由 Hello 报文中的优先级决定的。FHRP 会对特定对象的状态（接口、疏通情况等）进行监控，一旦发现故障，就通过一个叫作追踪的功能将当前活动路由器的优先级降低。这样，当监控对象发生故障时，FHRP 就会发出一个优先级已降低的 Hello 报文，促使故障转移启动。

  我们以下图所示的结构为例来具体看一下。在这个不进行追踪处理的结构中，即使 WAN 接口发生故障，FHRP 也不会执行故障转移，结果导致通信路径失谐。

  下面我们加入追踪处理后再来看看。FHRP 在对 WAN 接口状态的监控中发现了故障，于是发出一个优先级已降低的 Hello 报文。备用路由器收到该 Hello 报文之后，顺利升级为新的活动路由器。

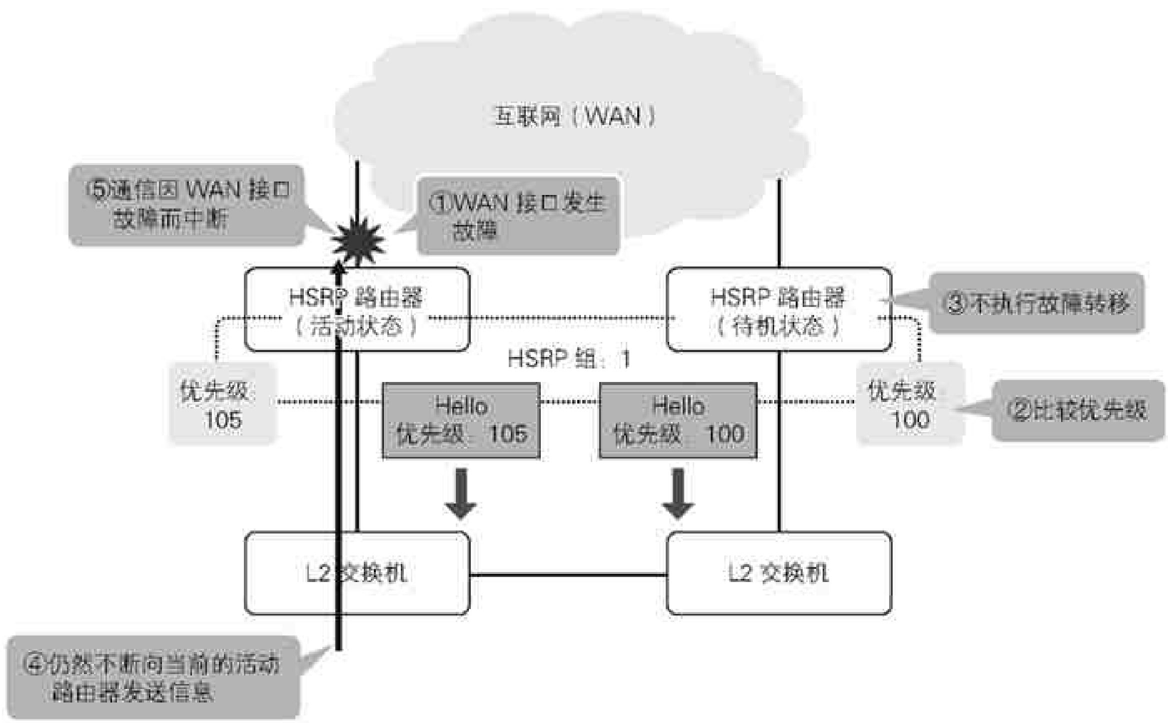


图 **4.1.57**　不进行追踪处理的话，通信路径就会失谐

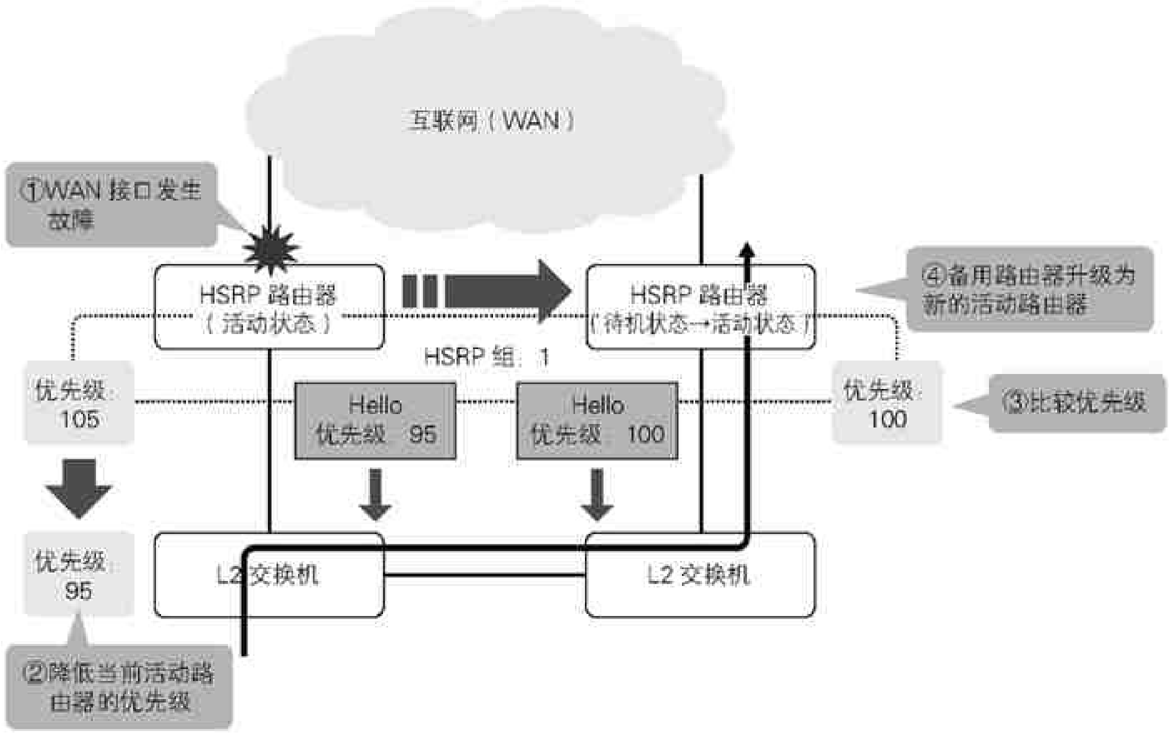


图 **4.1.58**　通过追踪处理强制启动故障转移

FHRP 的运作流程

接下来我们来看看 FHRP 是怎样实现活动 / 备用路由器的冗余结构的。这部分是 FHRP 的基本原理，HSRP 也好，VRRP 也好，运作流程都是一样的。而且，防火墙和负载均衡器的冗余备份采用的也是同样的机制，所以我们应该牢牢掌握这个部分。

ARP 是 FHRP 的基础技术和坚强后盾。人们通过对 ARP 的控制让通信流量集中到活动路由器（主路由器）上。在 FHRP 的结构中，活动 / 备用路由器各自拥有不同的物理 MAC/IP 地址，但与此同时，双方还拥有共享的虚拟 MAC/IP 地址。这个虚拟 MAC/IP 地址由服务器和 PC 的默认网关来设置。

- 正常情况下的运作流程

  首先我们来看看正常情况下的运作流程。

  1 →　针对发送给默认网关 IP 地址（即虚拟 IP 地址）的 ARP 请求，活动路由器会返回一个答复。回复的 MAC 地址即虚拟 MAC

地址。虚拟 MAC 地址因 FHRP 而异，HSRP 的虚拟 MAC 地址为 00-00-0C-07-AC-XX（这里的 XX 为组 ID），VRRP 的虚拟 MAC 地址则为 00-00-5E-00-01-XX（这里的 XX 为虚拟路由器 ID）。

2 →　在客户端的 ARP 条目中写入虚拟 MAC 地址和虚拟 IP 地址。

3 →　数据包被转发给活动路由器。

4 →　活动路由器根据本机中的路由表信息转发数据包。

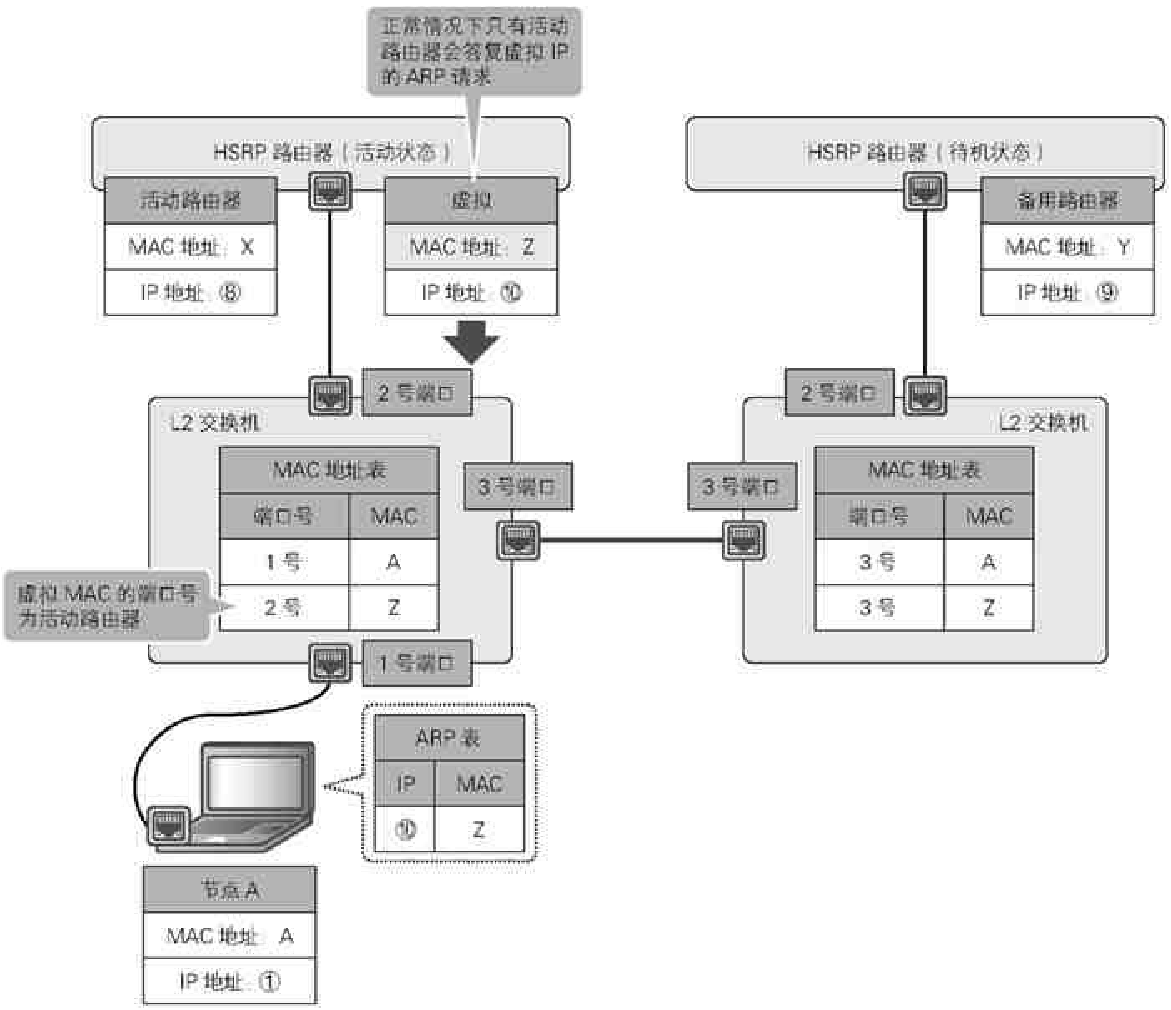


图 **4.1.59**　正常情况下只有活动路由器会答复虚拟 **IP** 的 **ARP** 请求

- 发生故障时的运作流程

下面我们再来看看启动故障转移时的运作流程。

1 → 活动路由器发生故障。

2 → 备用路由器未能收到来自活动路由器的 Hello 报文，或是收到了优先级较低的 Hello 报文，于是升级为新的活动路由器，同时发出 GARP 声明。

3 → GARP 更新 L2 交换机的 MAC 地址表，将通往虚拟 IP 地址的路径引向新的活动路由器。此时客户端的 ARP 条目并不更新，虚拟 IP 地址的 MAC 地址仍然保持虚拟 MAC 地址的原样不变。

4 → 数据包被转发给新的活动路由器。

5 → 新的活动路由器根据本机中的路由表信息转发数据包。

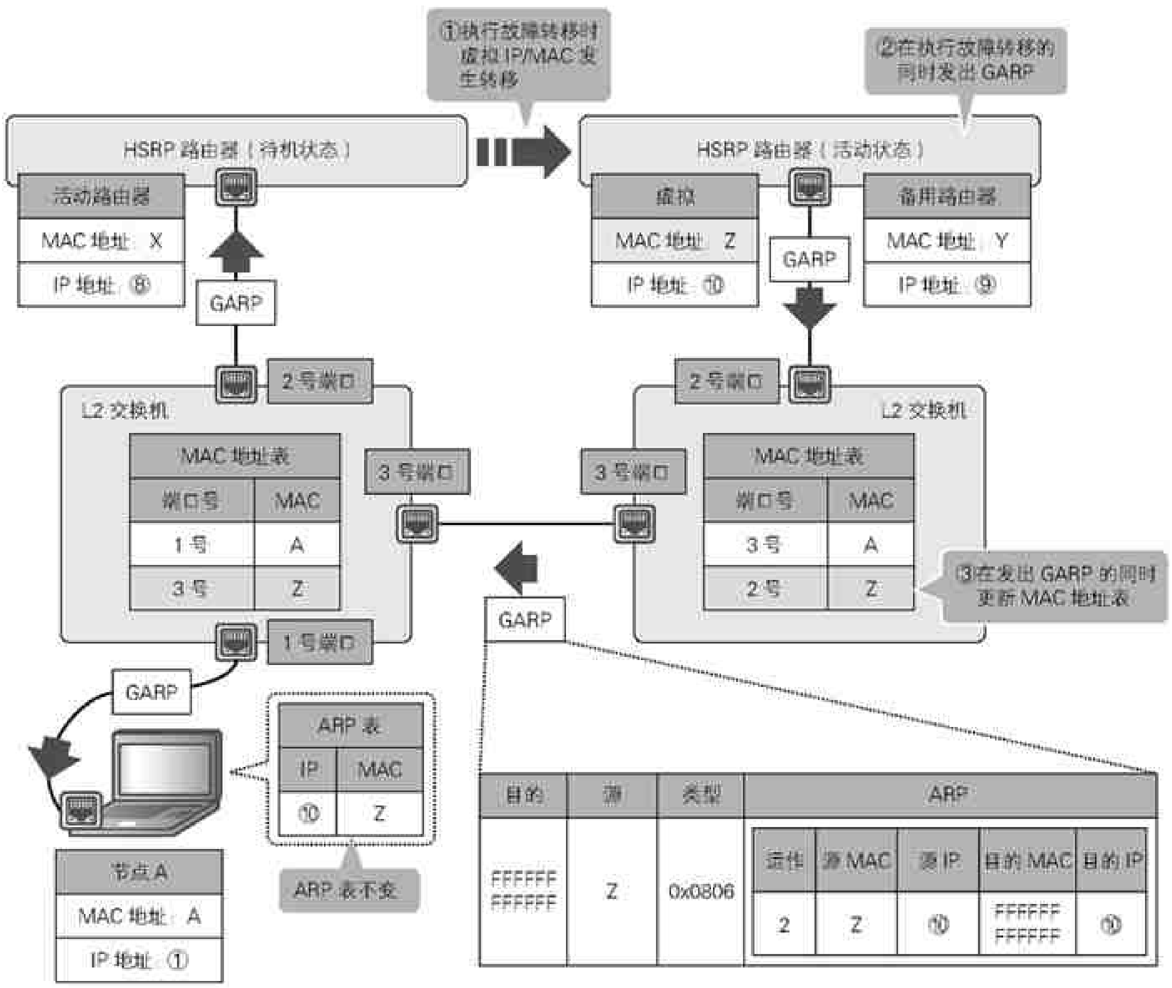


图 **4.1.60**　执行故障转移时新的活动交换机会发出 **GARP**

注意 **ID** 冲突

使用 FHRP 时，我们最应该注意的一点就是要避免 ID 发生冲突。FHRP 是仅凭 Hello 报文中的组 ID[10] 去识别 FHRP 组中的路由器的。因此，如果 FHRP 组外的路由器使用了和组内路由器相同的 ID，FHRP 就会无法正常运作。

> [10] 如果是 VRRP，则是依靠 Advertisement 报文中的虚拟路由器 ID 去识别 VRRP 组中的路由器的。

在连接构成 FHRP 的设备时，我们要做三道确认工作。首先，要看上层设备是否是用了 FHRP。如果回答是肯定的，要继续看使用的是哪些 FHRP。如果使用的是相同的 FHRP，就还要进一步确认它们分别使用的是什么 ID。当上层设备为数据中心的设施时，使用什么 ID 往往是有具体规定的，因此遇到这种情况，我们就要使用指定好的 ID。

让根网桥和活动路由器彼此对准位置

数据链路层通过 STP 实现冗余备份，网络层则通过 FHRP 实现冗余备份——这在不久之前还曾是绝对的业界标准，但是最近，这对冗余技术已经逐渐被 StackWise 技术和 VSS 所取代，今后人们在新建冗余备份时可能就不会再用到它们了。不过，既然它们仍将继续存在，那我们掌握好它们的相关知识总是有利无弊的。

通过 STP 和 FHRP 实现冗余备份的时候，我们必须让根网桥和活动路由器彼此对准位置。如果根网桥对准的是备用路由器，那么通信要到达上层设备就不得不绕远。绕远就会造成延迟，既耽误时间，也影响效率，还会给运行管理带来一些不必要的麻烦。因此，我们一定要遵守让 STP 的根网桥和 FHRP 的活动路由器彼此对准位置这条原则。

通过多个组 **ID** 来实现活动 **/** 活动冗余结构

FHRP 在每个组中采用的都是活动 / 备用冗余结构。我们不妨这样认为：因为每个组都有一个虚拟 IP 地址，所以每个虚拟 IP 地址都对应着一个活动 / 备用冗余结构。而 FHRP 也可以利用这种机制，架构出活动 / 活动冗余结构。具体来说，就是以虚拟 IP 地址为单位去切换活动机，

这样，从整体的角度来看就实现了活动 / 活动冗余结构。毋庸置疑，在这种架构中，我们必须想办法让下属客户端的默认网关能够得到合理的分散，否则活动 / 备用冗余结构也是无法工作的。

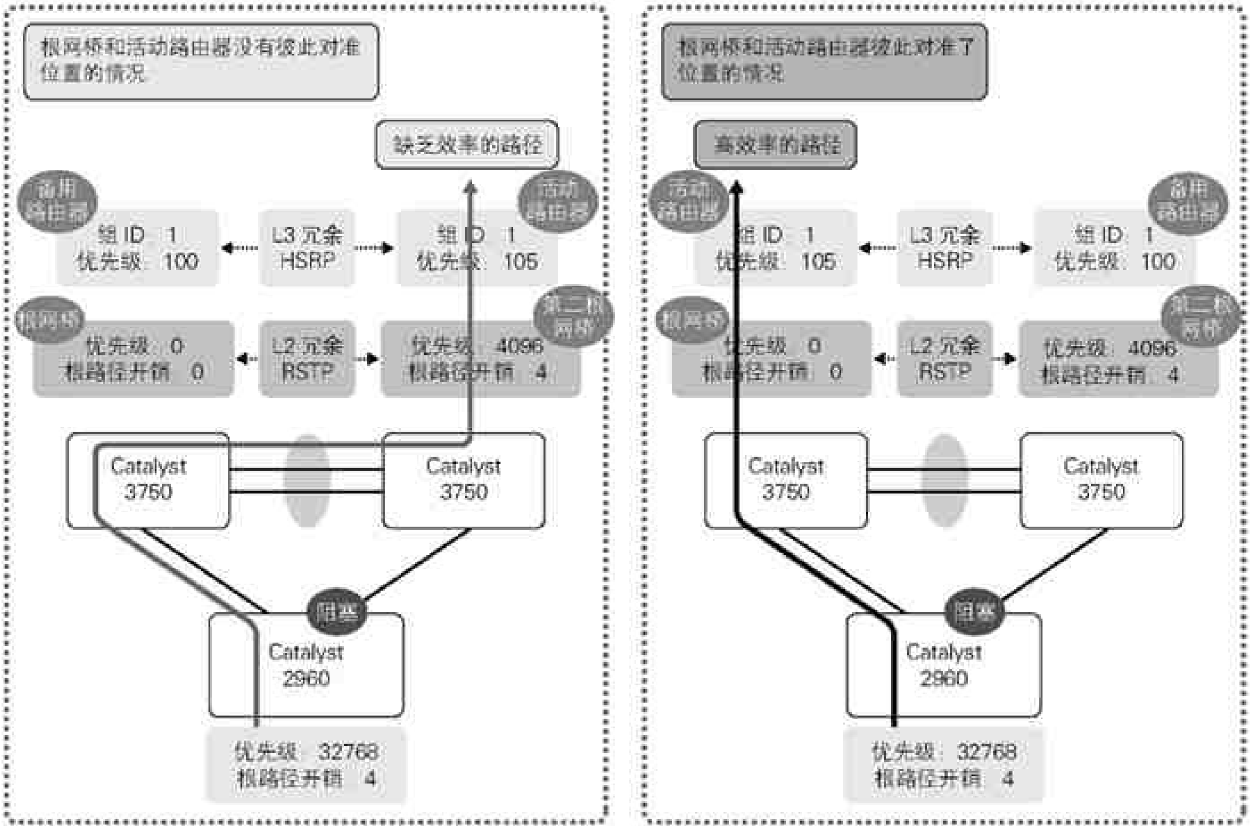


图 **4.1.61**　让根网桥和活动路由器彼此对准位置

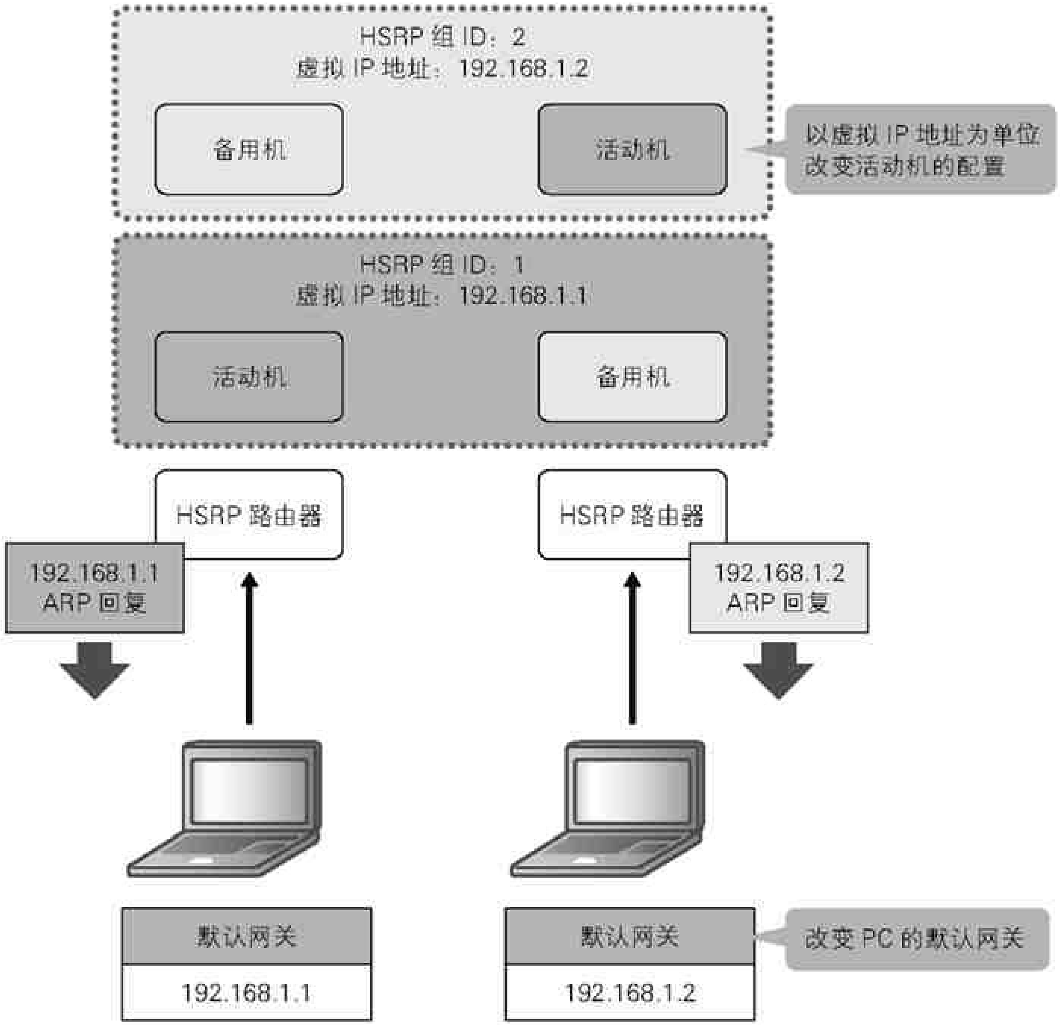


图 **4.1.62**　以虚拟 **IP** 地址为单位对活动机进行切换

最近几年，随着通信流量的激增，活动 / 备用冗余结构和 n ＋ 1 结构似有卷土重来之势。这大概是因为很多人认为，既然备用机大部分时间都在休眠，那么何不让它们发挥作用，去分散一部分通信流量呢？那样的话处理效率一定会更高……然而，我们必须清楚地认识到，基于 FHRP 的活动机 / 备用机冗余结构有着一个很大的缺点，那就是运行管理非常麻烦。如果你的确有与之相当的技术水平，非常善于管理，又相信自己能够完美地驾驭这种冗余结构，那么当然没有任何问题。不过，如果一发生问题你还是得求助于工程师的话，那么笔者还是劝你不要使用这种冗余结构了。

4.1.3.2　利用路由协议确保通往上层设备的路径

路由协议不仅负责学习路径信息，还肩负着实现冗余备份的重任。路由器和 L3 交换机将路径信息暂时保存在非路由表的表中，在真正的路由表中只写入度量值最小的那条最佳路径。路径学习告一段落之后，路由协议会敦促路由器发出 Hello 报文或 KEEPALIVE 消息以了解对方的状态，发生故障时也是利用这些数据包去检测故障，确保迂回路径。

通过 **BGP** 实现通往 **ISP** 的路径冗余

服务器端是通过 BGP 确保通往 ISP 的路径冗余的。BGP 在学习完路径之后会敦促路由器发出 KEEPALIVE 消息，以此来定期监控彼此的状态。

ISP 对于 KEEPALIVE 消息的发送间隔和 Hold 时间都会提示推荐时长，我们应按照这些提示去完成设置。这里的推荐时长可以理解为通信中断时间。另外，内部是通过 FHRP 实现活动 / 备用冗余结构的。因此，我们应将冗余结构下直属设备（在图 4.1.63 所示的结构中为防火墙）的默认网关设置为 FHRP 的虚拟 IP 地址。

如果在 Hold 时间内未能收到 KEEPALIVE 消息，BGP 就会断定已有故障发生，在历经多段路程之后另外开通出一条通往 ISP 的路径并确保其畅通无阻。由于 BGP 是通过 KEEPALIVE 消息判断出故障的，所以即使途中某处断掉也能够马上发现问题。另外，ISP 的路径即使被切换掉，FHRP 也不会执行故障转移，毕竟切换的只是上层路由信息而已。

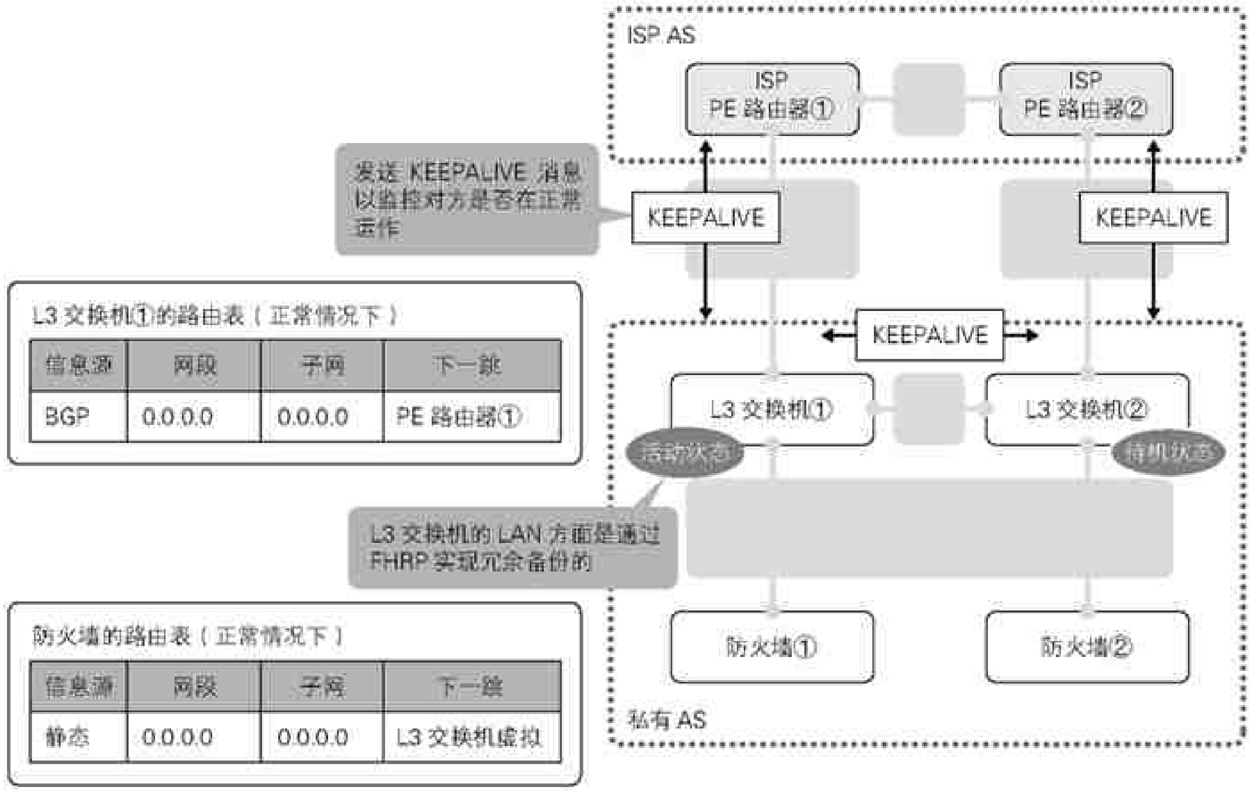


图 **4.1.63**　通过发送 **KEEPALIVE** 消息监控路径状态

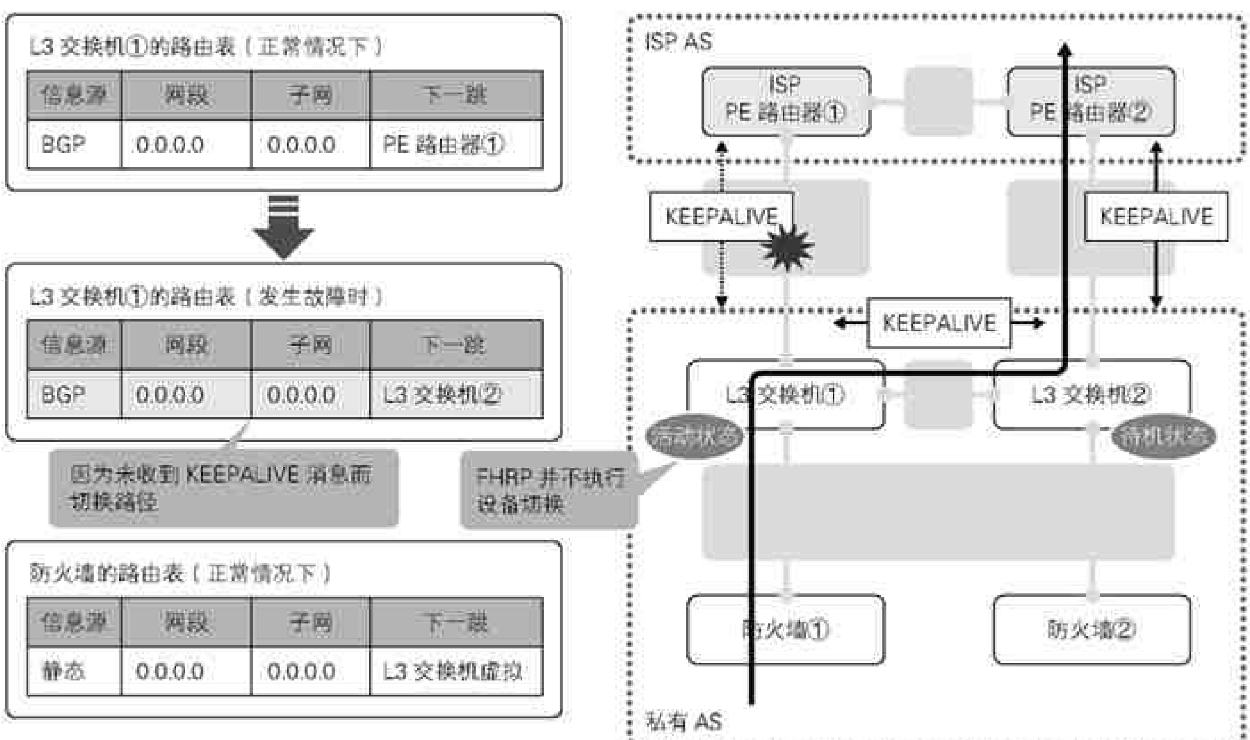
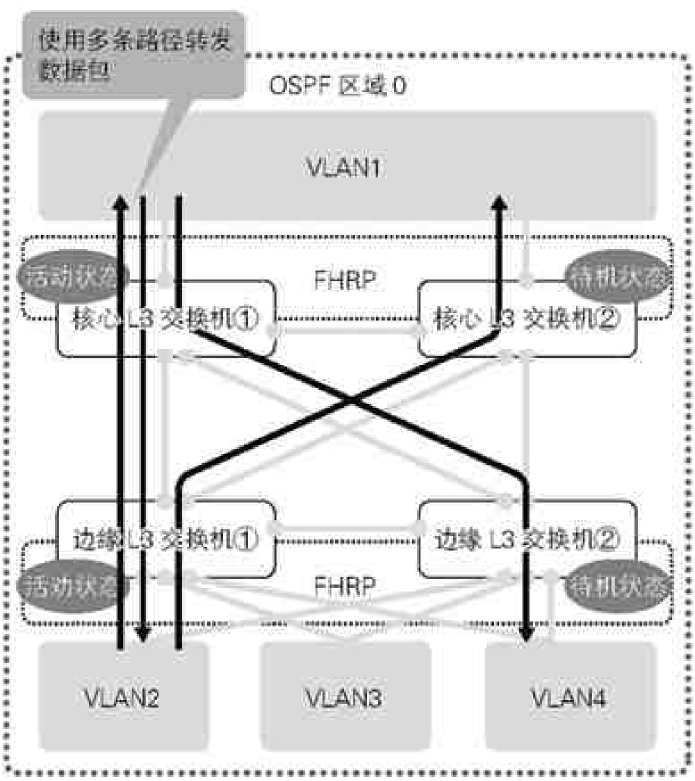


图 **4.1.64**　通过发送 **KEEPALIVE** 消息检测出故障

LAN 内的路径通过 **IGP** 实现冗余

LAN 内的路径是通过 OSPF 或 EIGRP 等 IGP 来实现冗余的。在不久之前的网络设计中，人们还倾向于将 VLAN 尽量做小并且大量地分布路由选择点。在这个前提下，通过 IGP 去实现路径冗余是必需的手段。OSPF 和 EIGRP 的默认设置均为等价多路径（度量值相同的路径全部都要用上），因此能够同时实现带宽增强和路径冗余。



信息源	目的网段	下一跳
OSPF	VLAN2	边缘 L3①
OSPF	VLAN2	边缘 L3②
OSPF	VLAN3	边缘 L3①
OSPF	VLAN3	边缘 L3②
OSPF	VLAN4	边缘 L3①
OSPF	VLAN4	边缘 L3②

信息源	目的网段	下一跳
OSPF	VLAN1	核心 L3①
OSPF	VLAN1	核心 L3②

图 **4.1.65**　通过 **IGP** 实现路径冗余

4.1.4　从传输层到应用层的冗余技术

从传输层到应用层的冗余技术在网络层的冗余技术 FHRP 中又加入了同步技术，从而实现了更上一层楼的效果。基本工作机制并没有变化，不过对于不同的信息会进行不同的同步处理，下面我们就来详细说明。

4.1.4.1　防火墙的冗余技术

防火墙的冗余技术是在 FHRP 中添加了几项同步技术，包括设置信息和连接信息的同步等，进而在更高的层次上实现了冗余备份。关于这两项同步技术，我们先来通过它们和 FHRP 的比较来说明一下。

同步设置信息

FHRP 中活动路由器和备用路由器的设置信息是各自独立的，并没有同步。因此，如果我们修改了活动路由器的设置，那么也要同样修改备用路由器的设置，否则就无法保持二者的一致性。与此相对，防火墙的冗余技术能够根据活动机的设置自动同步备用机的设置，让设备的运行管理更加方便快捷。

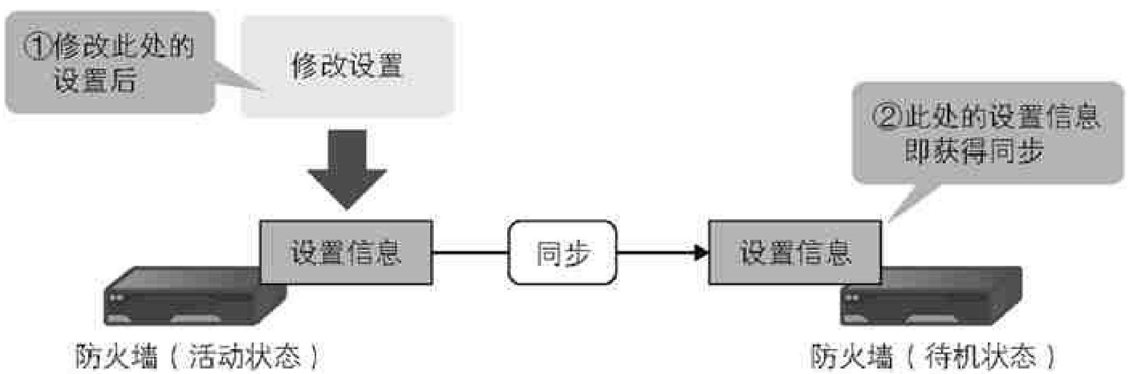


图 **4.1.66**　同步设置信息

同步连接信息

虽然 FHRP 会通过发送 Hello（Advertisement）报文去掌握对方的通信状态，但并没有对连接信息进行同步处理。说起来，数据包这一级的路径变更和连接、应用程序的运行本来就没有太大的关系，所以人们一般认为没必要去同步连接信息。但是防火墙就不一样了，防火墙是基于 TCP 连接信息来建立过滤规则的。从这个角度来看，不同步连接信息是说不过去的，因为假如连接中断而不得不重建，应用程序就会无法接通，这可称得上是重大事故了。为了避免出现这种情况，我们必须在防火墙中同步连接信息，给备用机预先建立起过滤规则，这样才能尽量减少发生故障时 TCP 层面的宕机时间。

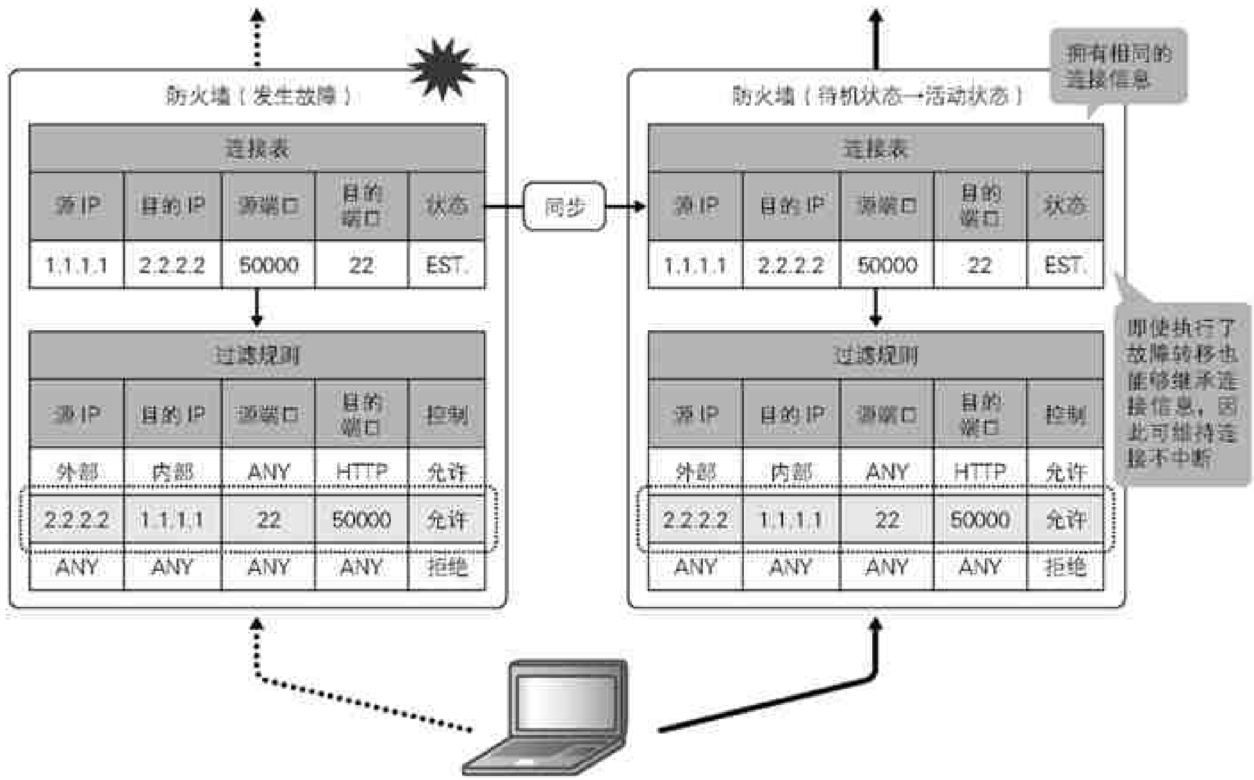


图 **4.1.67**　同步连接信息

用于同步的链路应另外设计

防火墙的冗余技术要求我们必须严格遵守一项原则，那就是要将用于同步的链路和用于同步的 VLAN 分别设计。用于同步的数据量可不像 Hello 报文或 Advertisement 报文那样少，并且由于是实时地同步信息，端口的 LED 灯会一直地猛烈闪动，简直让人怀疑是不是出现了环路故障。有鉴于此，我们应尽可能地避免同步与正常服务的通信并存，应将用于同步的链路和 VLAN 分开来设计。

用于同步的链路是防火墙冗余技术中最根本的一条重要链路，这条链路如果中断的话事态就非常严重了。为了避免出现这样的局面，人们一般会用链路聚合将链路捆绑在一起以应对原发性故障。有些设备还可以设置第二链路和第二 VLAN，以防主链路中断时没有备用的可供切换。对于这样的设备，请务必将它们的 Trust VLAN（LAN 方面的 VLAN）定义成第二位的性质。笔者见过误将 Untrust VLAN 定义成第二位性质的网络环境，按照这个错误设置来的话，当主链路中断时管理信息就会全部泄露出去，是万万不可的。请记住，至少，第二链路一定要定义成 Trust VLAN 中的链路才行。

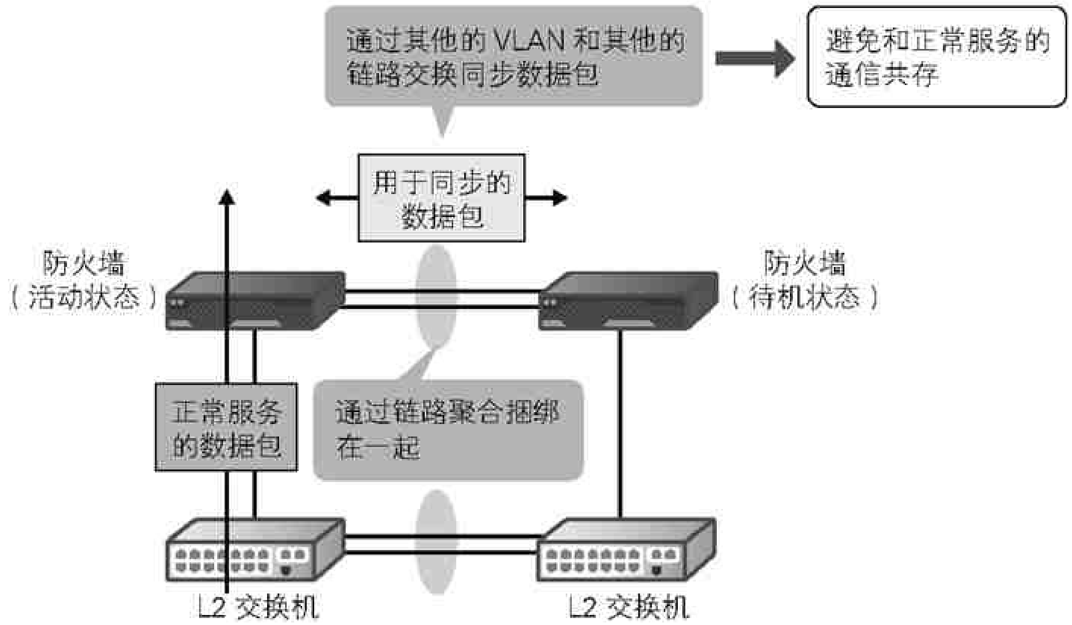


图 **4.1.68**　用于同步的链路应另外设计

设置虚拟 **MAC** 地址

有些设备可以将它们真实的 MAC/IP 地址直接用作两机共享的虚拟 MAC/IP 地址，思科公司的 ASA 系列设备就是一例。在这个系列中，最初定义的主机的真实 MAC/IP 地址即默认的虚拟 MAC/IP 地址。使用这种设备时，主机一旦被更换，虚拟 MAC 地址就会改变，最终导致通信中断。F5 Networks 公司的 BIG-IP 系列设备则是另一种情况。在这个系列中，两机并没有共享的虚拟 MAC 地址，而是将活动机的真实 MAC 地址用作虚拟 IP 地址的 MAC 地址。使用这种设备时，一旦执行故障转移，所有相邻设备的 ARP 条目都需要通过 GARP 进行切换，所以周边设备的状态变化比较大。

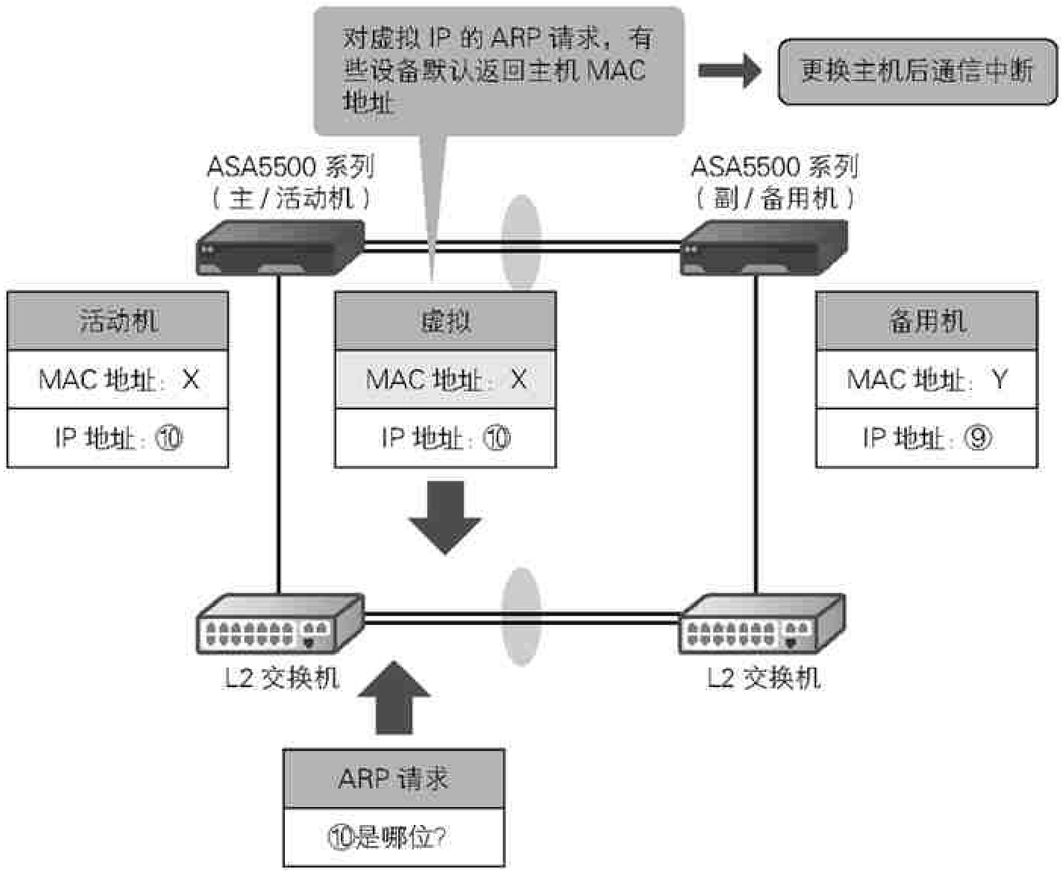


图 **4.1.69**　有些设备可将活动机的 **MAC** 地址直接用作虚拟 **MAC** 地址

如果设备本身允许指定虚拟 MAC 地址，那我们最好事先就设置好，这一点对任何设备的冗余备份来说都是一样的。因为这样的话，我们就能将相邻设备的状态变化控制在最小范围之内。

防火墙冗余技术的运作流程

请记住，防火墙冗余技术的基本运作流程和 FHRP 并没有太大的不同，只是在 FHRP 的基本运作中增加了同步处理而已。下面，我们就来看一看具体的步骤。

- 正常情况下的运作流程

  首先，我们来看看正常情况下的运作流程。

  1 →　针对发送给默认网关 IP 地址（即虚拟 IP 地址或 NAT 地址）的 ARP 请求，活动防火墙会返回一个答复。答复的 MAC 地

址即虚拟 MAC 地址。

2 →　在客户端的 ARP 条目中写入虚拟 MAC 地址和虚拟 IP 地址。

3 →　客户端中经由活动防火墙建立起 TCP 连接，具体来说是经过活动防火墙执行三次握手处理之后建立起 TCP 连接。

4 →　活动防火墙根据 TCP 连接信息建立起过滤规则，同时将该连接信息同步到备用防火墙中。于是备用防火墙也根据该连接信息建立起过滤规则。

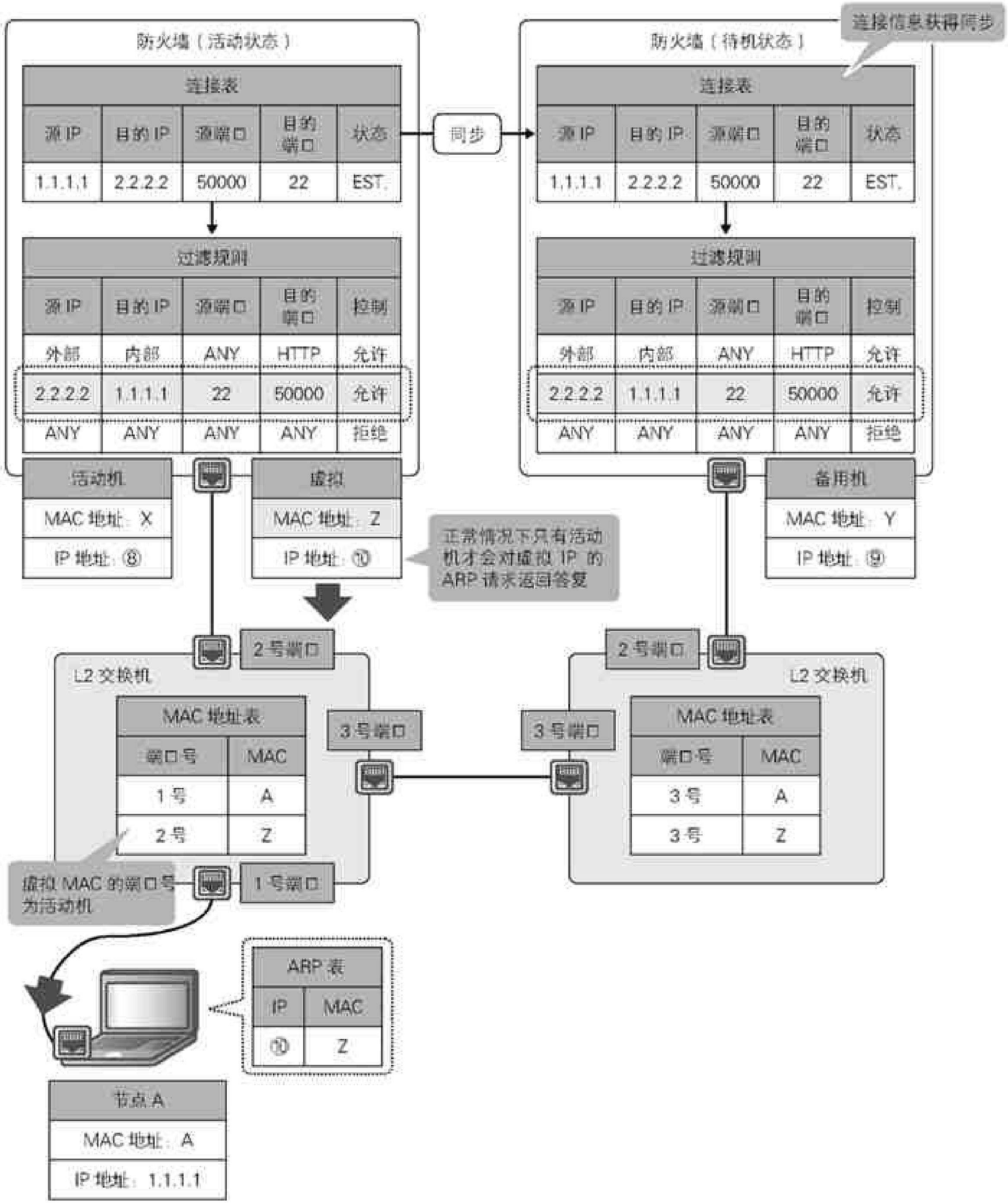


图 **4.1.70**　正常情况下的运作和 **FHRP** 基本一样

- 发生故障时的运作流程

接下来，我们再看看执行故障转移时的运作流程。

1 →　活动防火墙发生故障。

2 →　备用防火墙通过同步数据包检测到该故障并升级为新的活动防火墙，同时发出 GARP 声明。

3 →　GARP 更新 L2 交换机的 MAC 地址表，并将通往虚拟 IP 地址的路径引向新的活动防火墙。此时客户端的 ARP 条目并不更新，虚拟 IP 地址的 MAC 地址仍然保持虚拟 MAC 地址的原样不变。

4 →　经由新的活动防火墙建立连接。

5 →　由于新的活动防火墙中早已拥有同步好的连接信息，所以可继续保持连接状态。

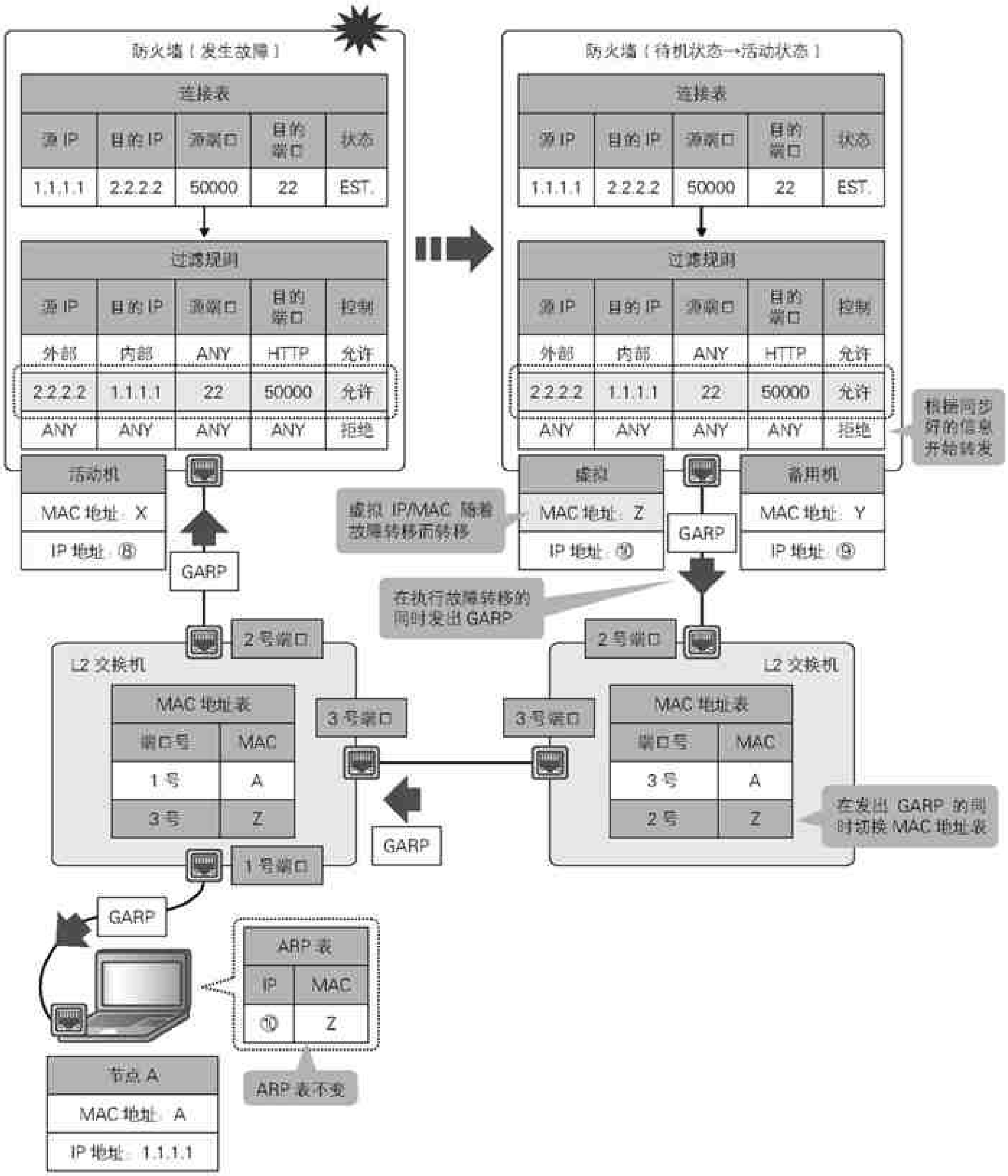


图 **4.1.71**　利用同步好的连接信息缩短通信中断时间

4.1.4.2　负载均衡器的冗余技术

负载均衡器的冗余技术是在防火墙冗余技术和同步技术的基础上进一步添加了应用程序的同步技术，能够在更高的层次上实现冗余备份。与防火墙的情况相比，二者的基本设计要点和冗余机制大都是一样的，只是

同步的范围稍有差异而已。这里，我们仅介绍一下负载均衡器冗余技术中重要的部分。

同步会话保持信息

负载均衡器冗余备份的关键在于同步会话保持表。会话保持是确保应用程序同步不可或缺的一项功能。执行故障转移的时候，如果我们没有同步会话保持表，应用程序就会被分配到另外的服务器中，导致前后无法呼应。因此我们必须始终同步会话保持的信息，使其能够持续不断地被分配到同一台服务器中，确保应用程序的前后呼应和一致性。

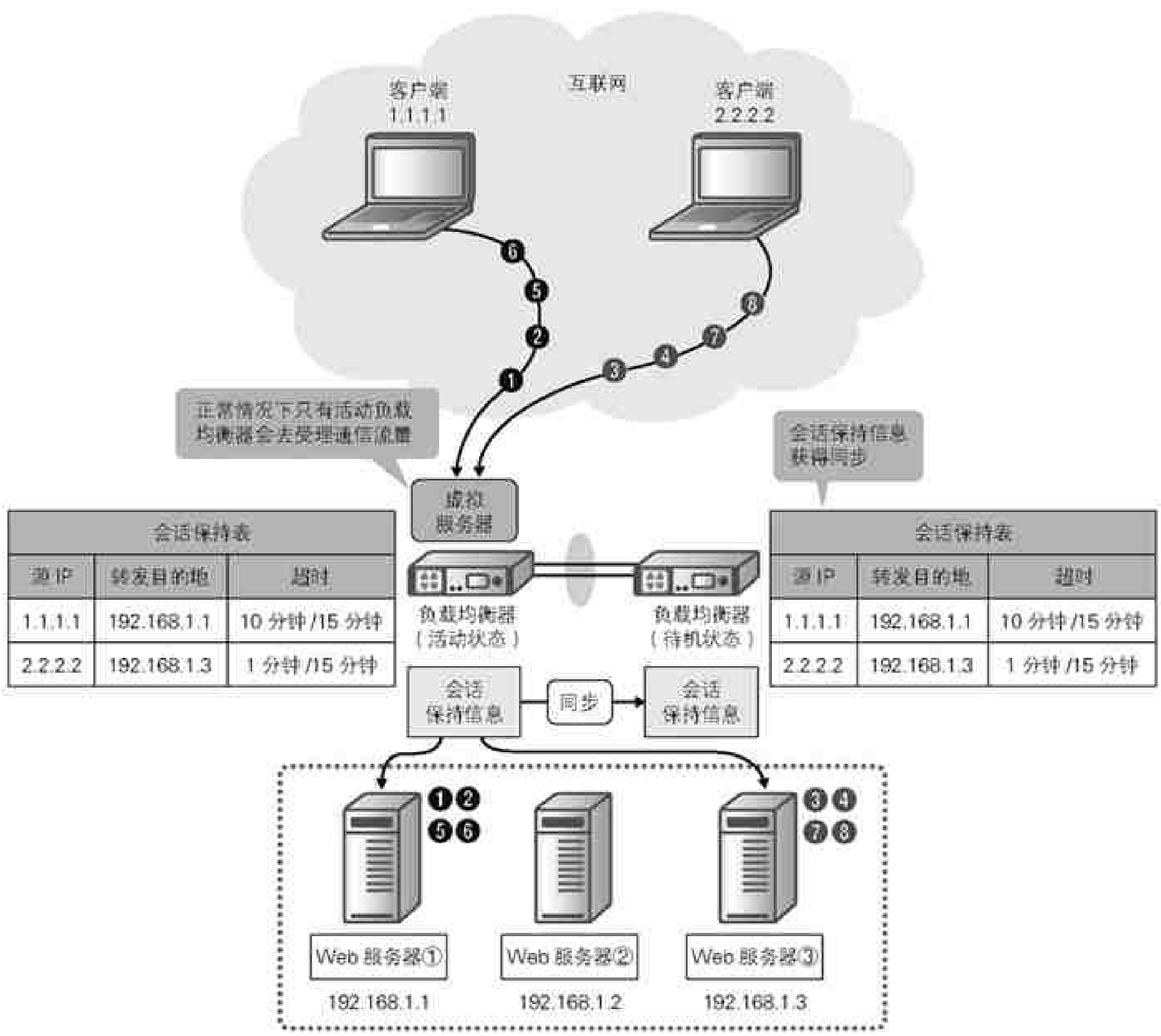


图 **4.1.72**　同步会话保持信息

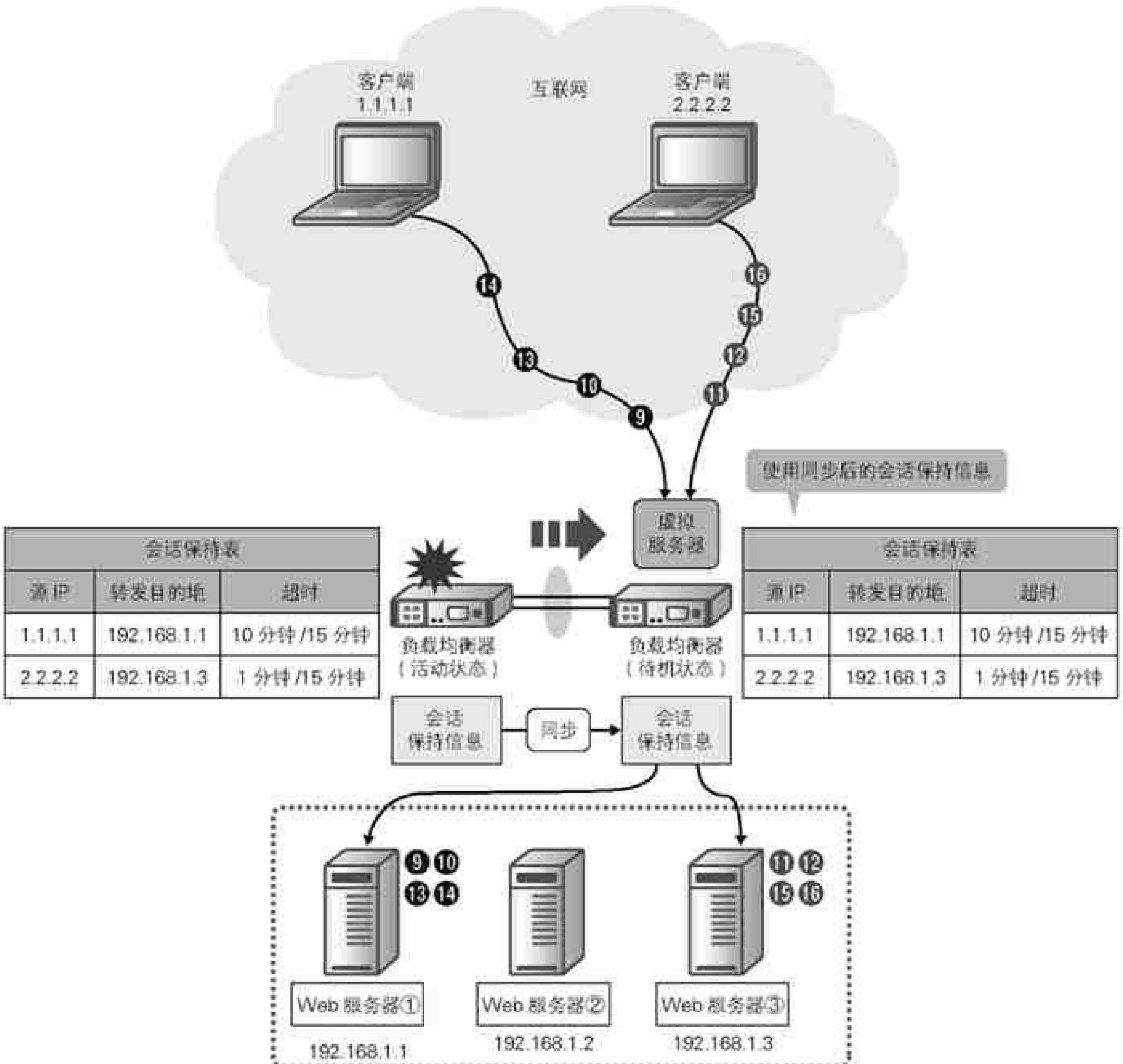


图 **4.1.73**　继承会话保持信息

4.2 高可用性设计

前面介绍了多种在冗余结构中使用的技术和这些技术的设计要点。从这一章节开始，本书将从实用性的角度来讲解如何在服务器端组合应用这些技术，以及我们在设计和架构网站时必须注意哪些事项。

4.2.1 高可用性设计

高可用性设计的要点在于并列配置。具体来说，就是将同一种类型的设备并列配置，然后用 L2 交换机将它们拼接起来。在第 1 章里已经讲过，冗余结构只有串联式和单路并联式两种。下面，就分别说明在这两种结构类型中分别利用了怎样的冗余技术，又是如何利用这些冗余技术的。

4.2.1.1 串联式结构

首先，我们来看看串联式结构。在串联式结构中，各个组成部分各司其职，结构非常简单。结构中的路径一目了然，故障也容易排除，因此深受管理人员的青睐。在介绍物理设计的章节中我们已经接触过几种结构实例，下面将基于这些实例来进行讲解。

串联式结构之类型 **1**

我们先来看看第一种类型（如图 4.2.1），这是串联式结构中最简单、最容易理解的一种网络结构。人们就是在这种结构的基础上对各种冗余技术进行了组合和应用，这样说就比较容易理解了吧。下面我们就从 ISP 的角度出发，逐一看看各个组成要素分别是怎样的情况。

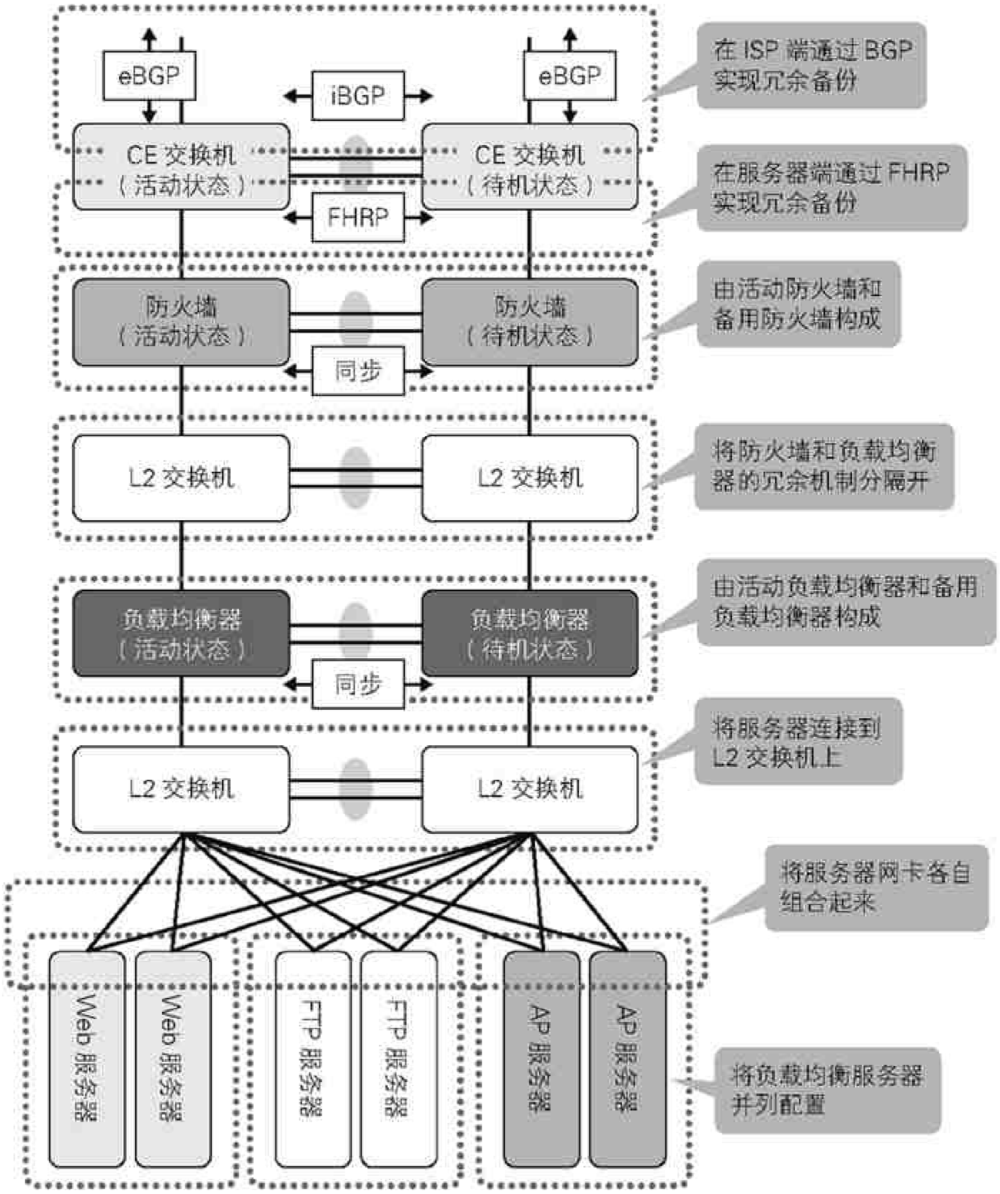


图 **4.2.1**　在串联式结构类型**1**中使用的冗余技术

- **CE** 交换机

CE（Customer Edge，客户边缘）交换机在互联网上和 LAN 上分别使用着不同的冗余技术。在互联网上是通过 BGP 实现冗余备份的。具体说来就是在 PE（Provider Edge，运营商边缘）交换机和 eBGP 对等体之间，以及 CE 交换机之间安排 iBGP 对等体，生成迂回路径。而在 LAN 上则是通过 FHRP 实现活动 / 备用冗余结构的。如果是出站通信，正常情况下只有活动路由器会接收数据包，然后根据通过 BGP 学习到的路径转发数据包；如果是入站通信，

则在 ISP 中操作 BGP 属性，让数据包只会被传给活动路由器。

事实上，由于 ISP 大多会提供租赁设备供我们使用，所以这部分的设置很可能并不需要我们亲自动手。不过，了解原理和机制对工程师来说是非常重要的，况且它们本身也非常有趣。

- 防火墙

防火墙同样也是由活动机和备用机构成的。我们应该另外设计用于同步的链路和用于同步的 VLAN，然后交换设置信息和状态信息的同步数据包，将设备状态的变化对服务通信流量的影响控制在最小范围之内。

- **L2** 交换机（设置在防火墙和负载均衡器之间）

设置在此处的 L2 交换机乍一看似乎毫无必要，确实常常会有系统管理人员过来询问“根本用不上吧？”然而回答是否定的。事实上，这台 L2 交换机起着非常重要的作用，它能将防火墙和负载均衡器的冗余机制分隔开。如果没有这台 L2 交换机，当防火墙或负载均衡器其中一方执行故障转移时，另一方就会受其影响也会随之执行故障转移。为了避免不必要的服务中断，这台 L2 交换机绝对是必要的。

- 负载均衡器

我们可以认为负载均衡器的设计和防火墙的基本上是一样的，同样也是由活动机和备用机构成，也是要另外单独设计用于同步的链路和用于同步的 VLAN，然后交换设置信息和状态信息的同步数据包，将设备状态的变化对服务通信流量的影响控制在最小范围之内。

- **L2** 交换机（设置在负载均衡器和服务器之间）

几乎没有任何一种结构是让服务器直接连接到负载均衡器上去的。说起来，负载均衡器本身就并不具备大量的物理端口，所以必须通过 L2 交换机去连接服务器。考虑到 L2 交换机本身可能会发生故障，所以应该连接成呈四方形的回路。

- 服务器

  服务器网卡是通过网卡组合实现冗余备份的。冗余方式有很多种，但最简单易行、也是最受人们欢迎的是由活动 / 备用机构成的 AFT 方式。考虑到交换机可能会发生故障，我们必须让活动机和备用机各自连接不同的交换机才行。采用 AFT 方式时，通信流量容易集中到活动网卡上，但是对于通信流量的增加，人们大多会选择增设负载均衡服务器这种横向扩展的办法来应对。横向扩展同时具有冗余备份的效果，人们将负载均衡服务器并列配置并实施健康检查，以此来实现服务器和服务本身的冗余备份。

串联式结构之类型 **2**

我们再来看第二种类型。这种类型是在第一种结构类型中加入了刀片服务器、虚拟化、StackWise 技术和 VSS、数据安全区域划分这四个元素（如图 4.2.2）。Untrust 区域和类型 1 是一样的。这里我们只选择性地介绍类型 2 和类型 1 不一样的地方。

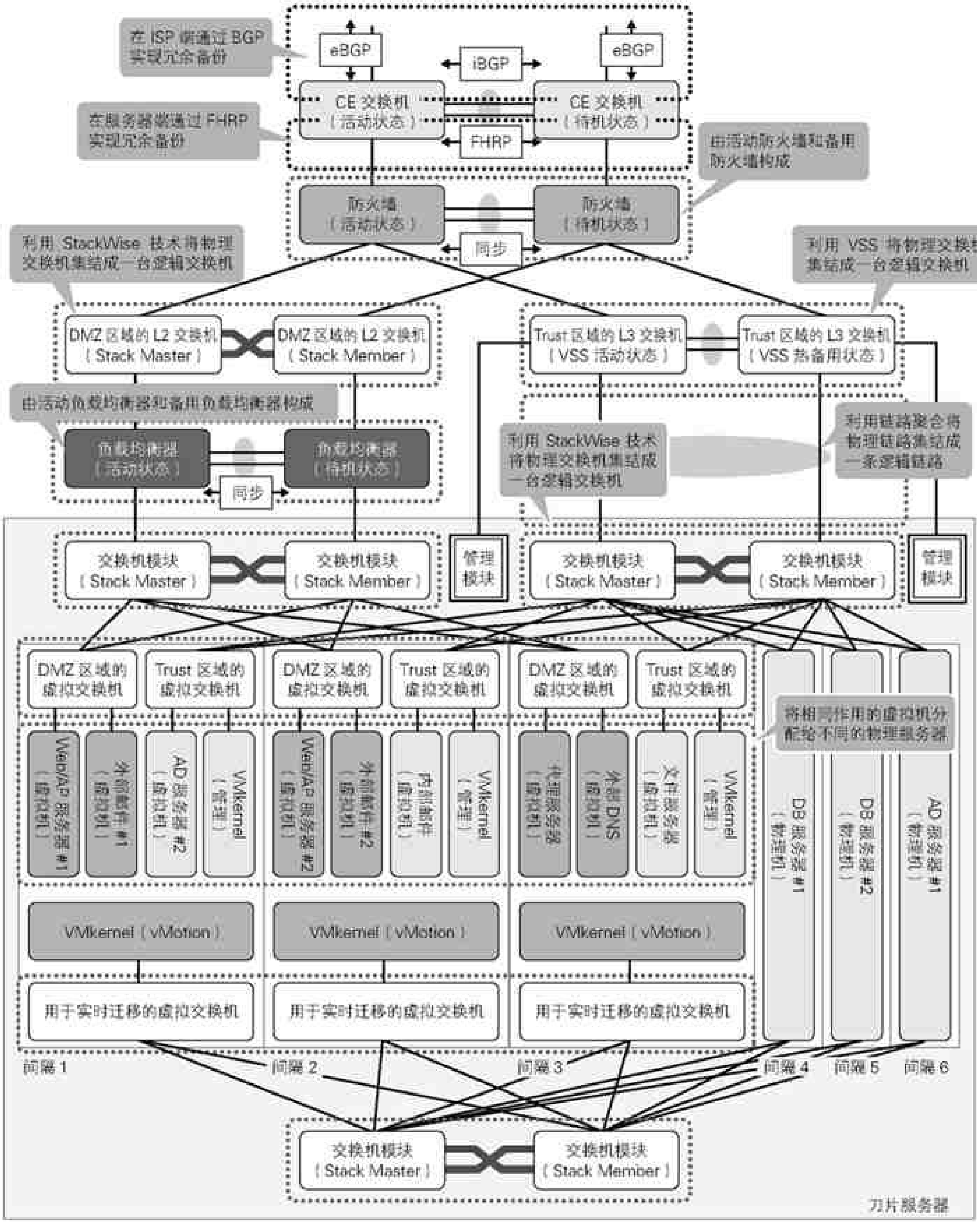


图 **4.2.2**　在串联式结构类型**2**中使用的冗余技术

※ 由于版面关系，部分服务器在图中省略未画。

- **L2** 交换机（**DMZ**）

  DMZ 区域的 L2 交换机利用 StackWise 技术使得冗余备份和结构简化能够同时实现。从物理角度上看存在两台交换机，但从逻辑角度上看则只有一台。不过，防火墙和负载均衡器应分别连接不同的物理交换机，以应对某台物理交换机发生故障的情况。

- 交换机模块

  交换机模块和 L2 交换机（DMZ）一样，也是通过 StackWise 技术实现冗余备份的。负载均衡器分别连接到不同的物理交换机上，以应对某台物理交换机发生故障的情况。至于 L3 交换机的连接，则是形成链路聚合之后分别从不同的物理交换机获取物理链路。交换机模块是按用途（DMZ、Trust、虚拟通信流量）划分开的，它们的通信流量不会互相影响。

- 刀片服务器和虚拟化处理

  刀片服务器按用途对应交换机模块，并与交换机模块是自动进行物理连线的，每个用途对应两台交换机模块，总共有六台交换机模块。利用连接好的网卡组合虚拟交换机，虚拟交换机组合的负载均衡多采用基于端口 ID 的方式或明确的故障转移方式，故障检测则多采用链路状态检测方式。虚拟化处理能够简单快速地添加服务器，这使得服务器容易在不知不觉中越来越多，所以我们必须巧妙地将负载均衡分配出去。另外，我们还得注意虚拟机的配置问题，如果将相同作用的虚拟机都分配给同一台物理机，那么该物理机一旦宕机，服务就会中断。因此，相同作用的虚拟机务必分配给不同的物理机才行。

- **L3** 交换机（**Trust** 区域）

  Trust 区域是通过 VSS 形成的。VSS 和 StackWise 技术一样，从物理角度上看存在两台交换机，但从逻辑角度上看则只有一台。两道防火墙应分别和不同的物理交换机连接，利用链路聚合使交换机模块的连接能够同时实现链路冗余和带宽扩展。分别连接不同的物理交换机，是为了让通信在某台交换机发生故障时不至于中断。

4.2.1.2 单路并联式结构

接下来，我们再看看单路并联式结构。在这种结构中，核心 L3 交换机扮演着多重角色，整体结构比串联式结构要复杂。不过，这种结构兼具逻辑结构的灵活性和可扩展性，非常适合规模较大的环境。单路并联式结构分为物理结构和逻辑结构两种，我们整理好这两种分类之后再去学习会比较容易理解一些。

单路并联式结构之类型 1

先来看第一种类型（如图 4.2.3），这是单路并联式结构中比较容易理解的一种网络结构。人们就是在这种结构的基础上对各种冗余技术进行了组合和应用，这样说就比较容易理解了吧。下面，我们就从 ISP 的角度出发，逐一看看各个组成要素分别是怎样的情况。

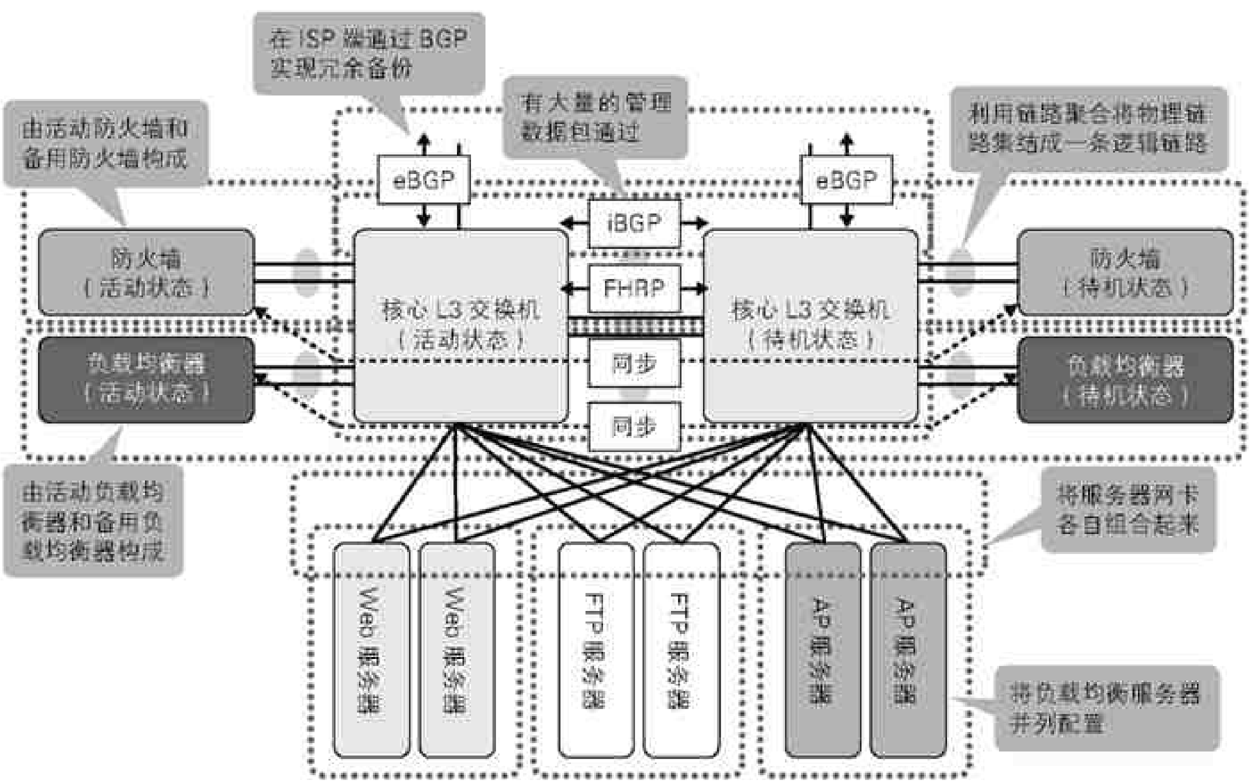


图 **4.2.3**　在单路并联式结构类型**1**中使用的冗余技术

- 核心 **L3** 交换机

  可以说，核心交换机决定了单路并联式结构的一切。ISP 的 PE 路由器之间的 BGP 对等体也要通过核心交换机去安排，LAN 内的 VLAN 通过 FHRP 形成活动 / 备用结构。在串联式结构中，为了减

少故障转移的影响我们非得配置 L2 交换机不可，然而在单路并联式结构中就不需要 L2 交换机了，因为核心交换机完全可以起到相同的作用。由于身兼数职，核心交换机发生故障时的影响还是比较大的，不过它能够统一管理复杂的端口分配工作，这是个很大的优点。

- 防火墙

防火墙由活动机和备用机构成，通过链路聚合连接核心 L3 交换机，能够同时实现冗余备份和带宽扩展。另外，防火墙是通过主干链路去连接多个 VLAN（包括 Untrust 区域、Trust 区域、用于同步的 VLAN 等）的。

用于同步的通信流量在通过链路聚合形成的逻辑链路中通行。由于通常服务和同步处理的通信流量同时并存，所以为了安全起见，有些情况下我们可以用 QoS 去控制优先顺序，或者干脆另外开设一条同步处理专用的链路，以尽量减少同步处理给通常服务造成的不良影响。

- 负载均衡器

我们可以认为负载均衡器的设计和防火墙基本上是一样的，同样也是由活动机和备用机构成，也是通过链路聚合连接核心 L3 交换机连接，以便同时实现冗余备份和带宽扩展。

用于同步的通信流量也是一样。用于同步的通信流量在通过链路聚合形成的逻辑链路中通行。由于通常服务和同步处理的通信流量同时并存，所以为了安全起见，有些情况下我们可以用 QoS 去控制优先顺序，或者干脆另外开设一条该 VLAN 专用的链路，以尽量减少同步处理给服务造成的不良影响。

- 服务器

在串联式结构中有专门连接服务器的交换机，而在单路并联式结构中起着同样作用的则是核心交换机。毋庸多言，每台服务器都必须分别连接不同的核心交换机，以应对某台核心交换机发生故障的情况。

单路并联式结构之类型 **2**

我们再来看第二种类型。这种类型是在第一种单路并联式结构类型中加入了刀片服务器、虚拟化、StackWise 技术和 VSS、数据安全区域划分这四个要素（如图 4.2.4）。这里我们只选择性地介绍类型 2 和类型 1 不一样的地方。

- 核心交换机

  核心交换机利用 VSS 实现冗余备份。从物理的角度上看有两台交换机，但从逻辑的角度上看只有一台，管理因此得以简化。就作用而言它和类型 1 并无太大差别，结构类型和类型 1 一样，也是核心交换机扮演多重角色，几乎决定了这种结构的一切。

- 交换机模块

  交换机模块通过 StackWise 技术实现冗余备份。至于核心交换机的连接，则是形成链路聚合之后分别从不同的物理交换机获取物理链路。交换机模块是按用途（DMZ、Trust、虚拟通信流量）划分开的，它们的通信流量不会互相影响。

- 刀片服务器和虚拟化处理

  刀片服务器按用途对应交换机模块，并与交换机模块是自动进行物理连线的，每个用途对应两台交换机模块，总共有六台交换机模块。利用连接好的网卡将虚拟交换机组合起来，虚拟交换机组合的负载均衡多采用基于端口 ID 的方式或明确的故障转移方式，故障检测则多采用链路状态检测方式。虚拟化处理能够简单快速地添加服务器，这使得服务器容易在不知不觉中越来越多，所以我们必须巧妙地将负载均衡分配出去。

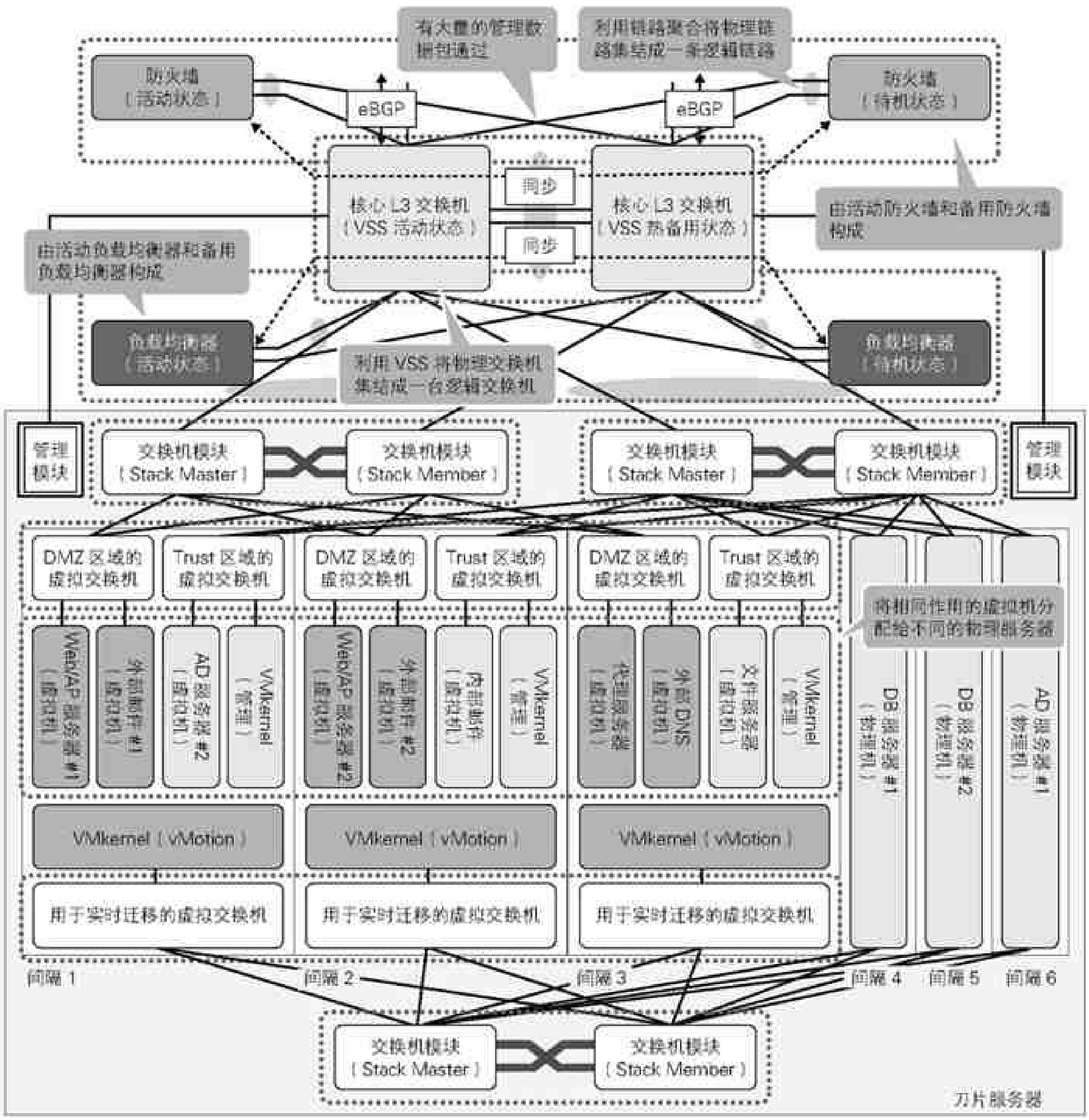


图 **4.2.4**　在单路并联式结构类型**2**中使用的冗余技术

4.2.2　理清通信流

笔者不止一次地被系统管理人员问到过“说了那么多，通信到底会通过哪里？又是怎样流动的呢？”这个问题将在下文中给出回答。不过这里介绍的通信流只是部分示例，实际情况不同，物理结构和逻辑结构也会有所差异。大家应做好充分的故障测试，便于自己理解通信流的知识。

服务器端的通信流分为串联式结构和单路并联式结构两种，二者之间的

区别很大，分开说明会比较容易理解一些。下面，本书将均以前面提到的结构类型 1 为例来一一说明。

4.2.2.1 串联式结构

串联式结构比较容易预测到发生故障的路径，故障排除也比较简单。活动路径的某处一旦断开就会立刻切换到备用路径，工作原理十分简单易懂。那么，下面我们就从 ISP 的角度出发，逐一看看当活动路径发生故障时，该结构中的各个组成要素是如何处理的。

- 正常情况下的路径

  我们先来看正常情况下的通信路径。在正常情况下通信流量一定是集中在活动机上的，这一点前面已经讲过了。这时候通信会流经下图中左侧的设备。

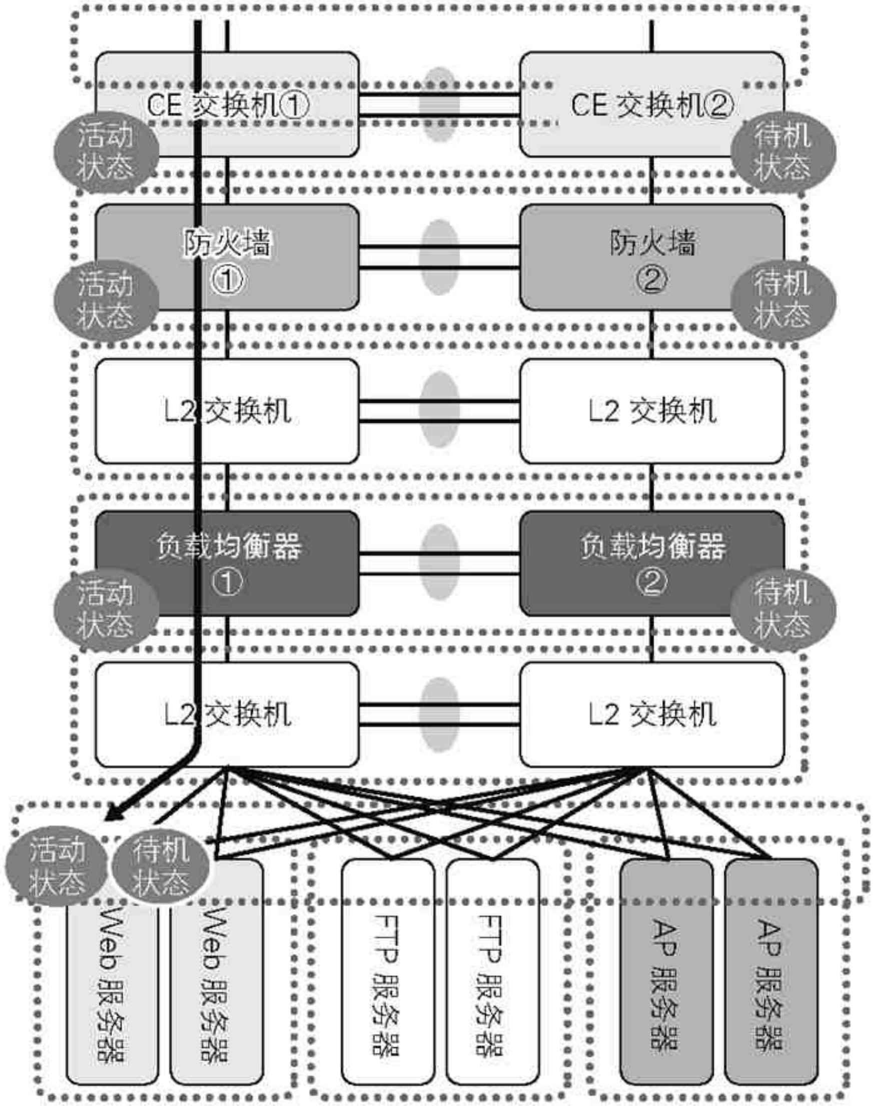


图 **4.2.5**　正常情况下通信只会流经活动机

- 线路故障

当 ISP 线路发生故障时，eBGP 对等体会因为收不到 KEEPALIVE 消息而中断，于是 ISP 网和 CE 交换机重新进行 BGP 计算，切换通往分配 IP 地址的路径。等计算告一段落之后，通往 ISP（通往互联网）的路径就会切换成经过配置在同一列的 CE 交换机（下图中的 CE 交换机②）的路径。

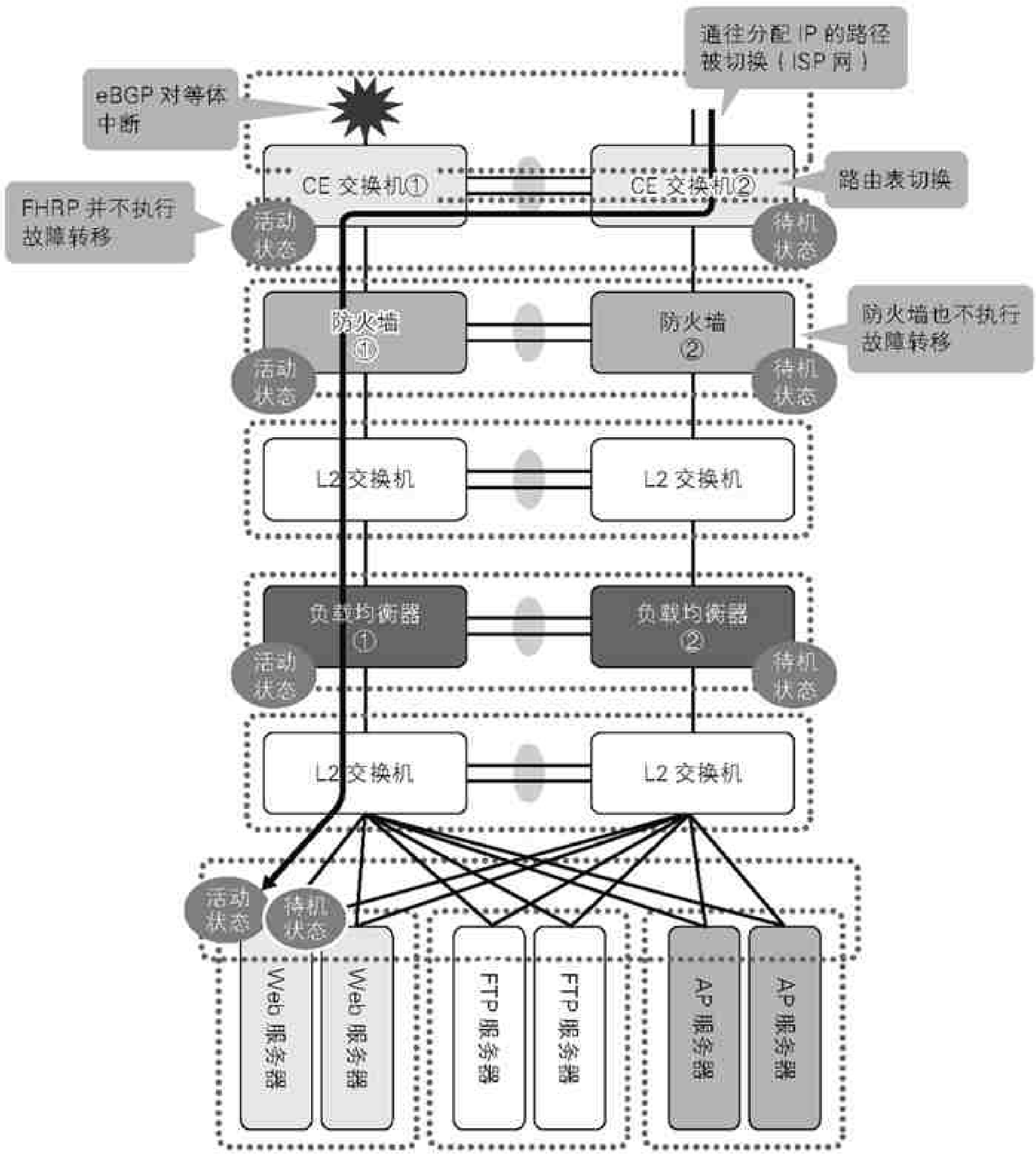


图 **4.2.6**　线路发生故障时会通过 **BGP** 切换路径

- **CE** 交换机故障

当 CE 交换机发生故障时，eBGP 对等体会因为 PE 路由器收不到 KEEPALIVE 消息而中断，于是 ISP 网和 CE 交换机重新进行 BGP 计算，切换通往分配 IP 地址的路径。此外，由于备用 CE 交换机内部收不到 Hello 报文，FHRP 会执行故障转移。

在这种结构中，防火墙同样也会执行故障转移。防火墙一直监控自身的链路状态，并将其作为故障转移的触发器。CE 交换机发生故障时，Untrust 区域的链路也会断掉，因此防火墙也会随之执行故障转移。

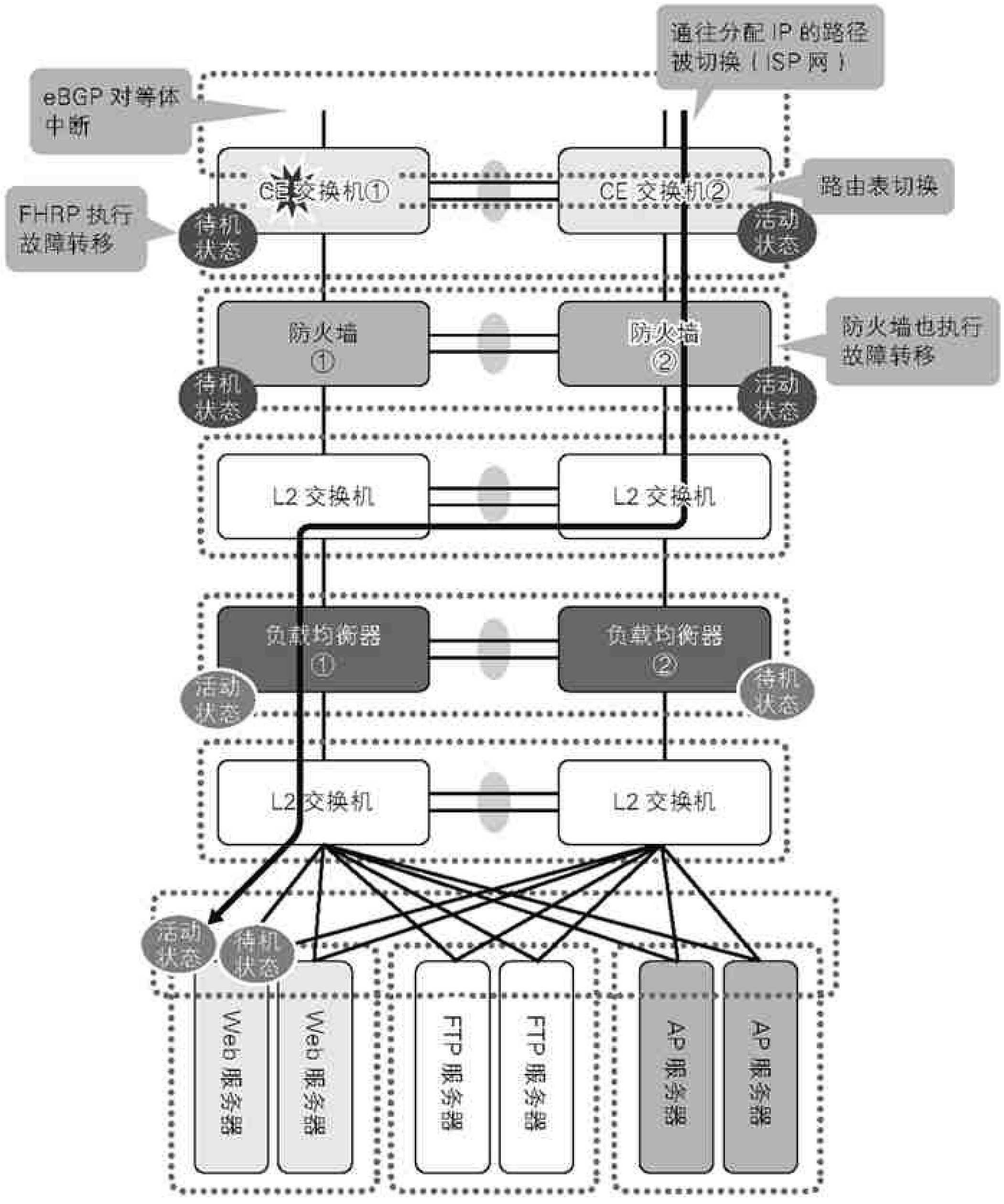


图 **4.2.7**　**CE** 交换机发生故障时，**CE** 交换机和防火墙都会执行故障转移

- **CE** 交换机内部链路（防火墙的 **Untrust** 区域链路）故障

在这种结构中，当 CE 交换机内部链路发生故障时，防火墙的 Untrust 区域链路也会随之中断。防火墙检测到链路中断后立即执行故障转移，而在 CE 交换机内部运作的 FHRP 则维持原状，并不执行故障转移。

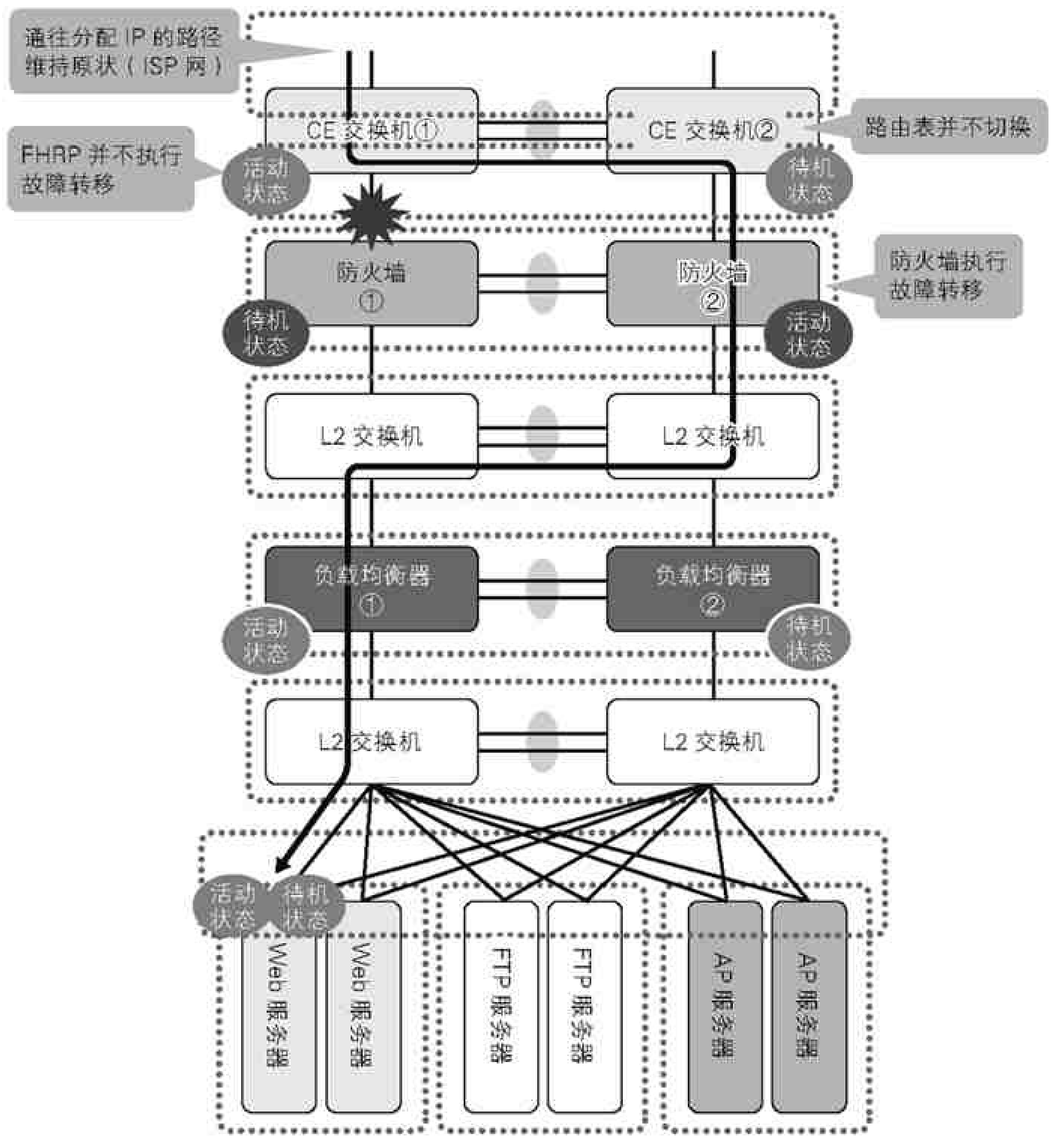


图 **4.2.8　CE** 交换机内部链路发生故障时，只有防火墙会执行故障转移

- 防火墙故障

当防火墙发生故障时，防火墙会检测到自身的故障并立即执行故障转移。在这种结构中，CE 交换机的内部链路也会因故障而中断，

但 FHRP 的 Hello 报文依然能在 CE 交换机之间的逻辑链路上正常交换，因此 FHRP 并不需要执行故障转移。

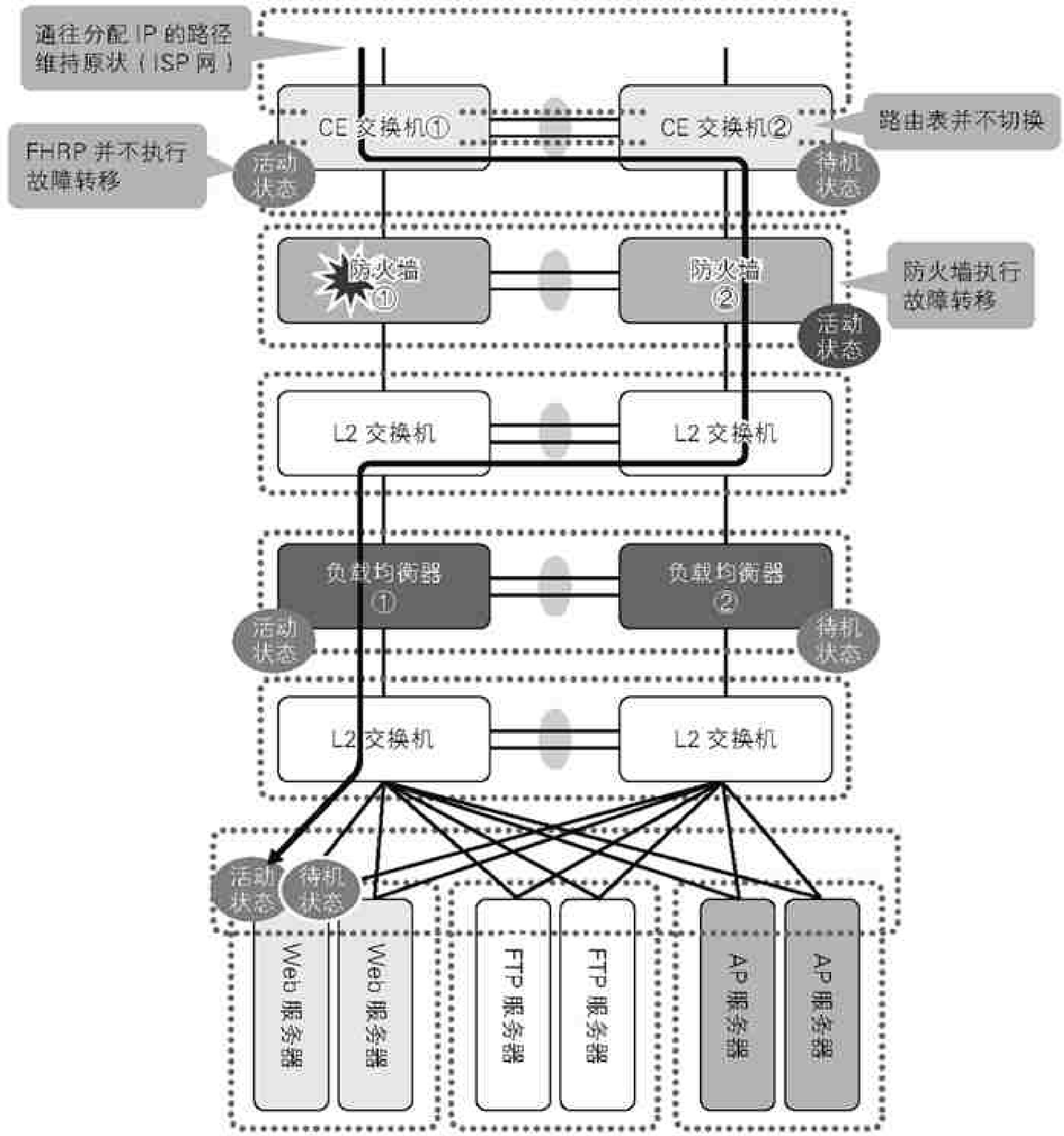


图 **4.2.9**　防火墙发生故障时，只有防火墙会执行故障转移

- 防火墙的 **Trust** 区域链路故障

当防火墙的 Trust 区域链路发生故障时，防火墙会检测到该故障并立即执行故障转移，其他部分则维持原状，不会执行故障转移。与此同时，L2 交换机会生成一条迂回路径，以避免防火墙的故障转移给负载均衡器造成不良影响。

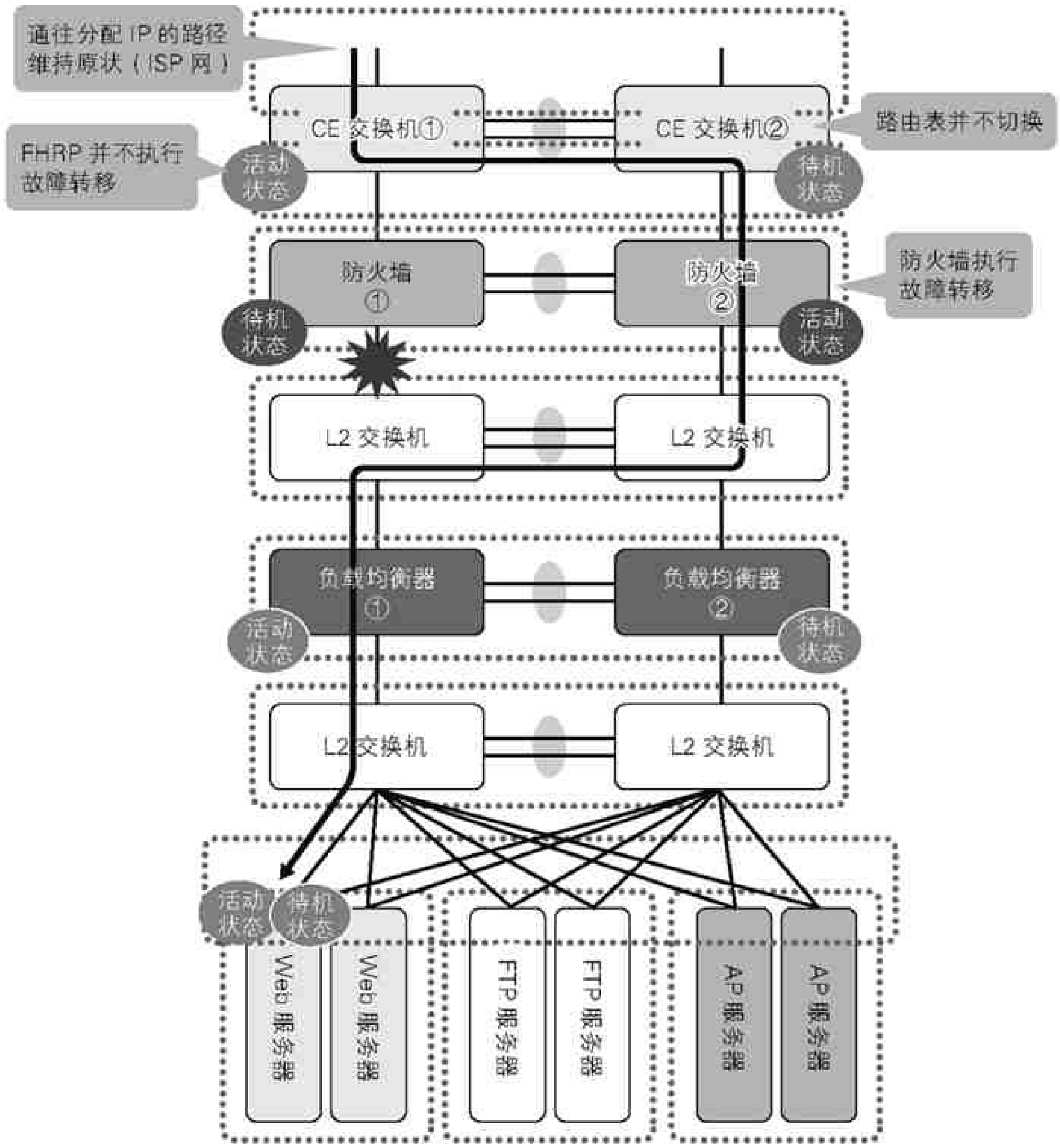


图 **4.2.10**　防火墙的 **Trust** 区域链路发生故障时，只有防火墙会执行故障转移

- **L2** 交换机故障（位于防火墙和负载均衡器之间）

当 L2 交换机发生故障时，防火墙和负载均衡器都会执行故障转移。防火墙通过该故障检测到 Trust 区域链路的中断，于是执行故障转移。

负载均衡器也会执行故障转移。负载均衡器和防火墙一样，一直在监控自身的链路状态，并将其作为故障转移的触发器。L2 交换机

发生故障时，负载均衡器的外部链路也会断掉，因此负载均衡器也会随之执行故障转移。

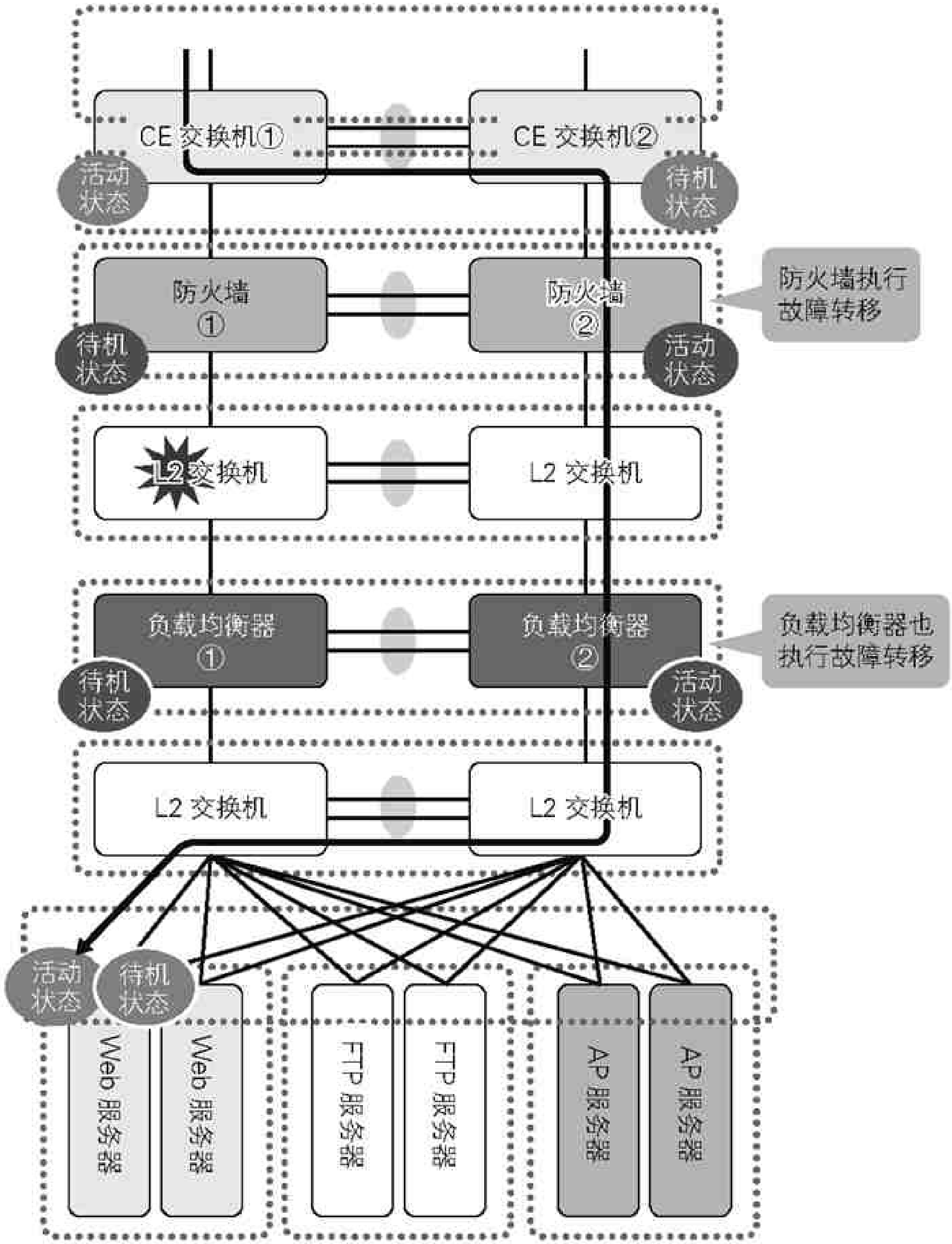


图 **4.2.11 L2** 交换机发生故障时，防火墙和负载均衡器都会执行故障转移

- 负载均衡器的外部链路故障

当负载均衡器的外部链路发生故障时，备用负载均衡器会检测到该故障并立即执行故障转移，其他部分则维持原状，不会执行故障转移。与此同时，L2 交换机会生成一条迂回路径，以避免负载均衡器的故障转移给防火墙造成不良影响。

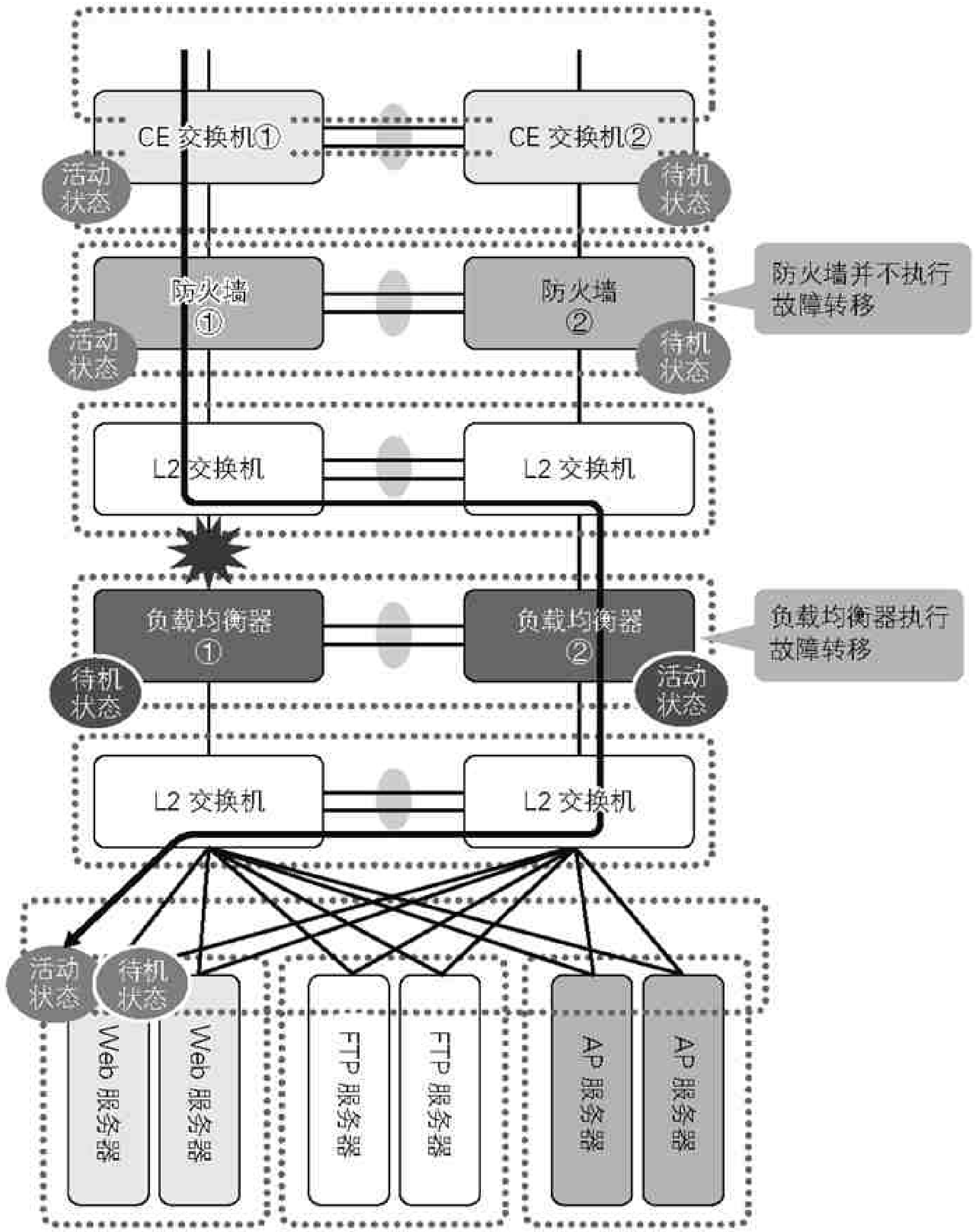


图 **4.2.12**　负载均衡器的外部链路发生故障时，只有负载均衡器会执行故障转移

- 负载均衡器故障

发生这种故障时，对应机制和负载均衡器的外部链路发生故障时的机制基本上是一样的。当负载均衡器发生故障时，备用负载均衡器会检测到该故障并立即执行故障转移，其他部分则维持原状，不会执行故障转移。与此同时，L2 交换机会生成一条迂回路径，以避免负载均衡器的故障转移给防火墙或服务器造成不良影响。

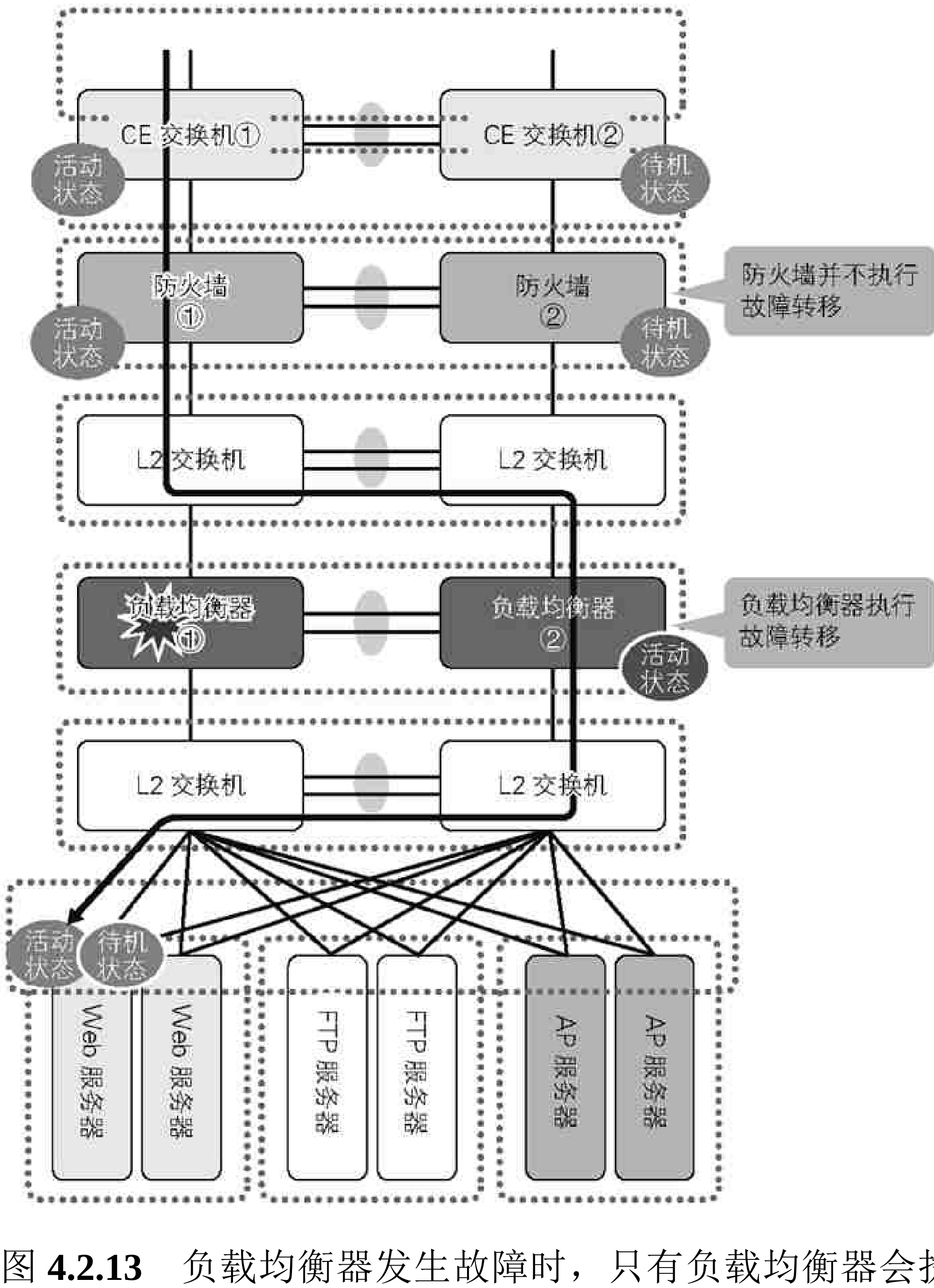


图 **4.2.13**　负载均衡器发生故障时，只有负载均衡器会执行故障转移

- 负载均衡器的内部链路故障

  发生这种故障时，对应机制也和负载均衡器的外部链路发生故障时的机制基本上是一样的。当负载均衡器的内部链路发生故障时，备用负载均衡器会检测到该故障并立即执行故障转移，其他部分则维持原状，不会执行故障转移。与此同时，L2 交换机会生成一条迂回路径，以避免负载均衡器的故障转移给防火墙或服务器造成不良影响。

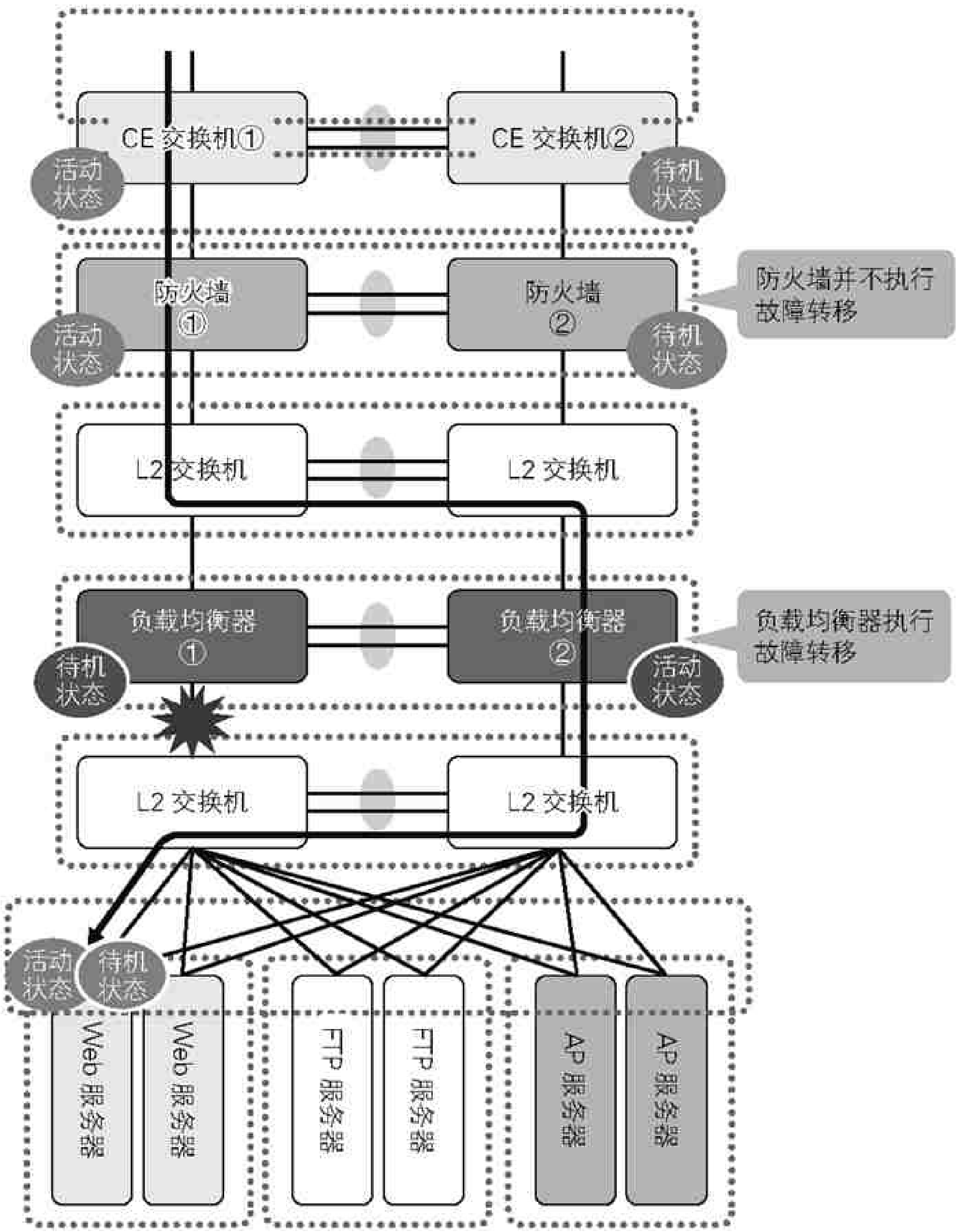


图 **4.2.14**　负载均衡器的内部链路发生故障时，只有负载均衡器会执行故障转移

- 内部的 **L2** 交换机故障（位于负载均衡器和服务器之间）

当内部的 L2 交换机发生故障时，不仅负载均衡器的内部链路会中断，连服务器的链路也会中断，因此负载均衡器和网卡组合都会执行故障转移。与此同时，L2 交换机会生成一条迂回路径，以避免负载均衡器的故障转移给防火墙带来不良影响。

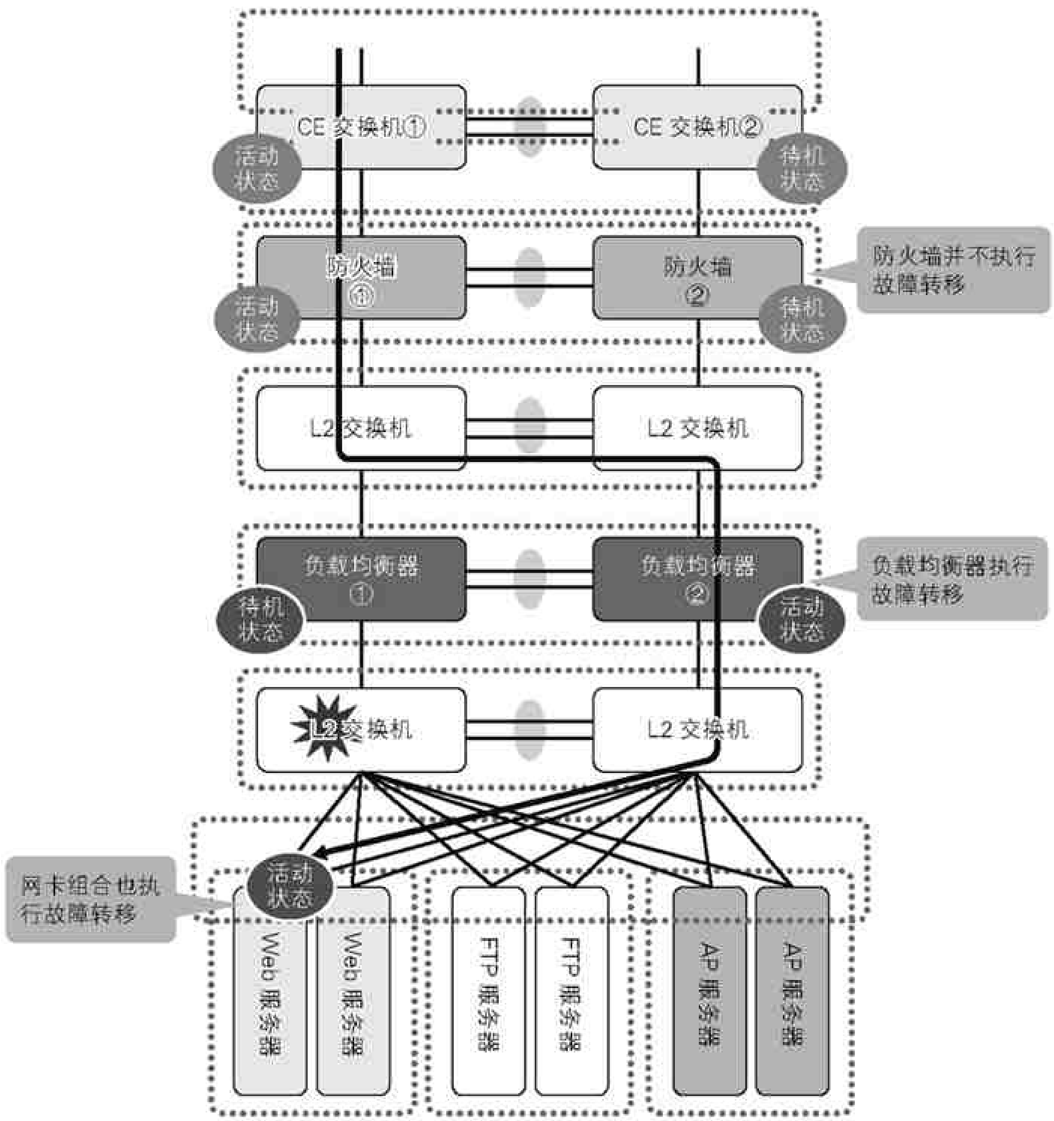


图 **4.2.15**　内部的 **L2** 交换机发生故障时，负载均衡器和网卡组合都会执行故障转移

- 服务器的链路故障和网卡故障

当服务器的链路或网卡发生故障时，只有网卡组合会执行故障转移。与此同时，L2 交换机会生成一条迂回路径，以避免网卡组合的故障转移给负载均衡器造成不良影响。

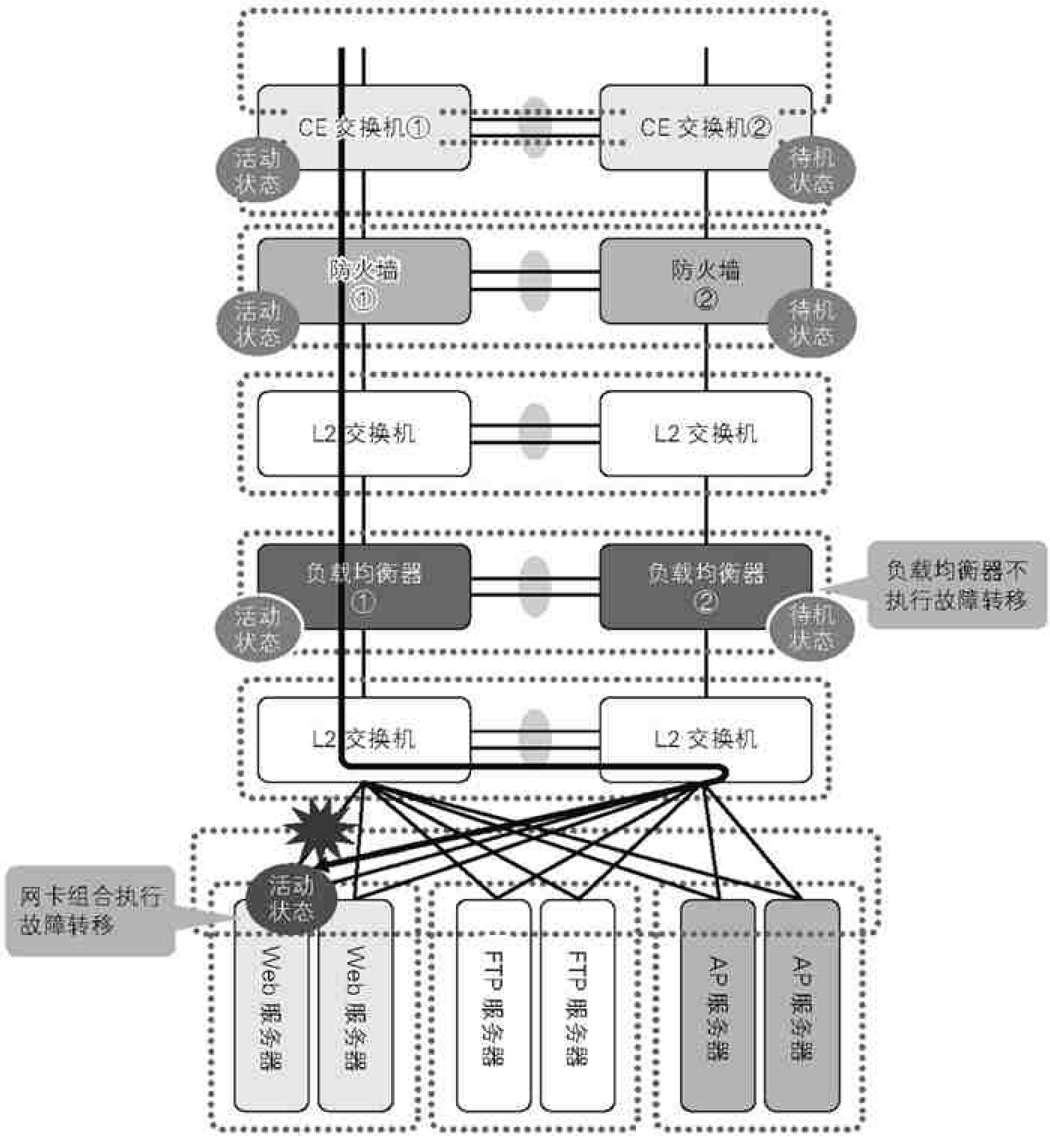


图 **4.2.16**　服务器的链路或网卡发生故障时，只有网卡组合会执行故障转移

- 服务器的服务故障

当服务器的服务发生故障时，负载均衡器的健康检查会失败，于是

该服务器被排除到负载均衡服务器范围之外，这样连接就不会被分配到已发生服务故障的服务器上。其他部分没有任何变化。

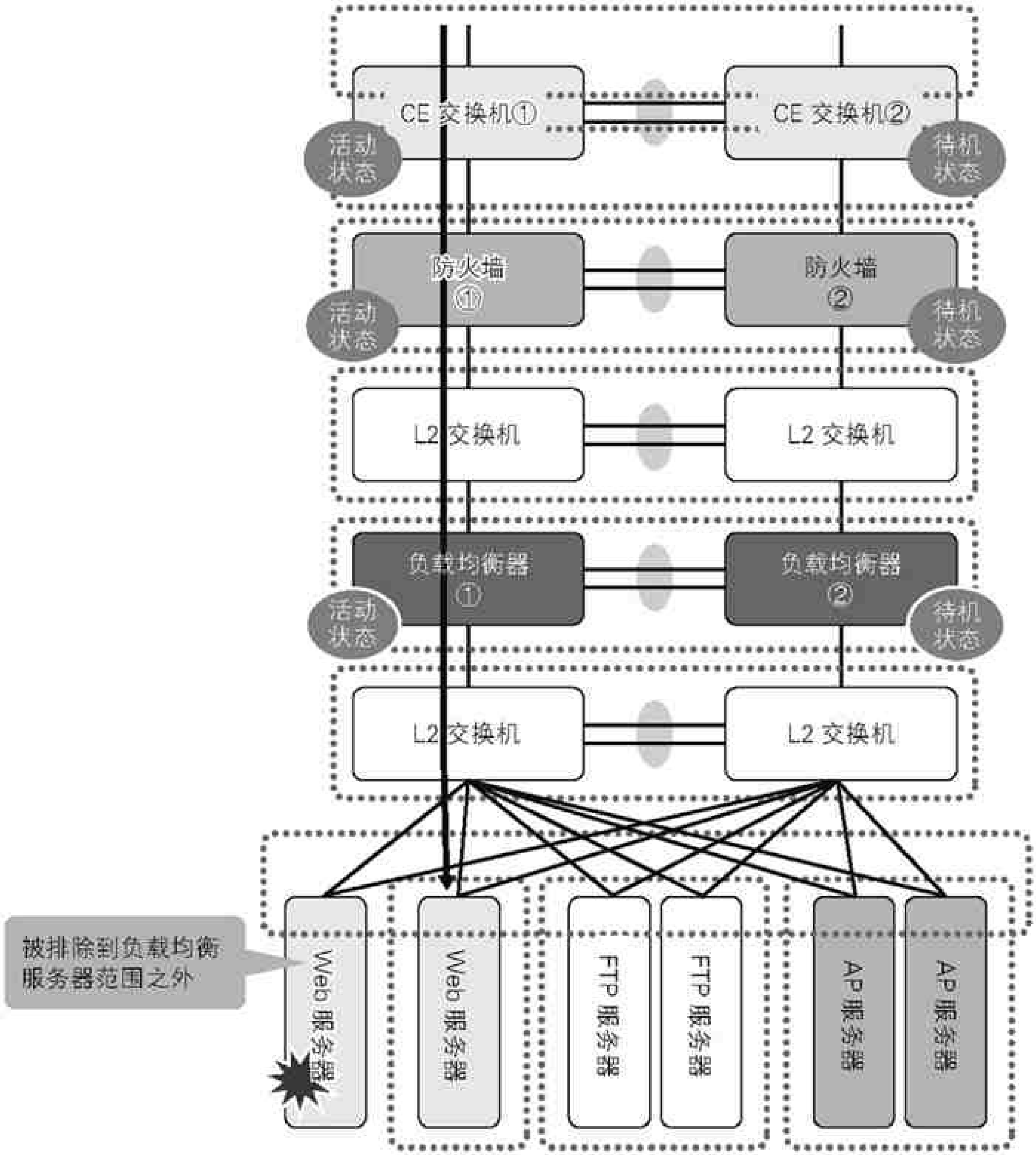


图 **4.2.17**　服务器的服务发生故障时，该服务器会被排除到负载均衡服务器范围之外

- 非活动路径的故障

如果是非活动路径发生故障，那么无论故障发生在什么环节都不会执行故障转移，也不存在通信中断时间。不过从笔者的经验来说，偶尔也会有这样的情况：因某些网络设备本身的设计漏洞等原因导

致其发送了不必要的 GARP，结果离奇宕机。为了避免类似情况的发生，我们应提前做好充分的故障测试，确保各设备都不会受到不良影响。

4.2.2.2　单路并联式结构

在单路并联式结构中，核心 L3 交换机扮演着多重角色，通信流似乎也比串联式结构要复杂得多。不过实际上它们的逻辑原理是一样的，所以用不着有太多的思想负担。活动路径的某处一旦发生故障，就应切换成备用路径——我们应该本着这个原则，从 ISP 的角度出发逐一排除故障。

- 正常情况下的路径

  我们先来看正常情况下的通信路径。在正常情况下，通信流量一定是集中在活动机上的，这一点前面已经讲过了。这时候通信会流经下图中左侧的设备。和串联式结构不同的是，单路并联式结构的通信会多次经过核心交换机，这是因为所有交换机（包括 CE 交换机和 L2 交换机等）的任务都要由核心交换机去完成的缘故。至于逻辑路径，单路并联式结构和串联式结构都是一样的，二者只是物理路径不同而已。

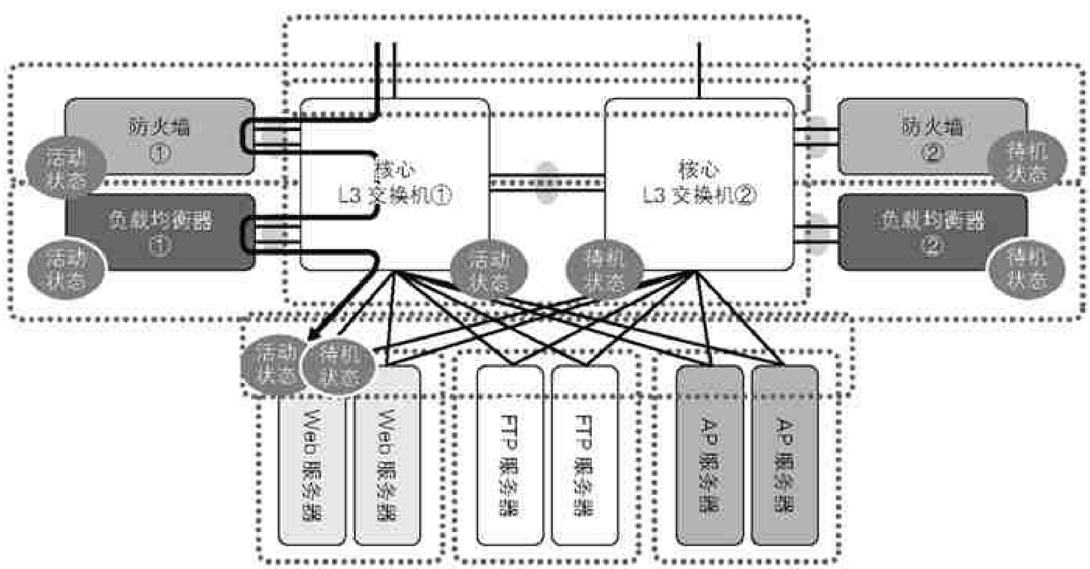


图 **4.2.18**　正常情况下，通信只会流经活动机

- 线路故障

当 ISP 线路发生故障时，eBGP 对等体会因为收不到 KEEPALIVE 消息而中断，于是 ISP 网和核心交换机重新进行 BGP 计算，切换通往分配 IP 地址的路径。等计算告一段落之后，通往 ISP（通往互联网）的路径就会切换成经过配置在同一列的核心交换机（下图中的核心交换机②）的路径。

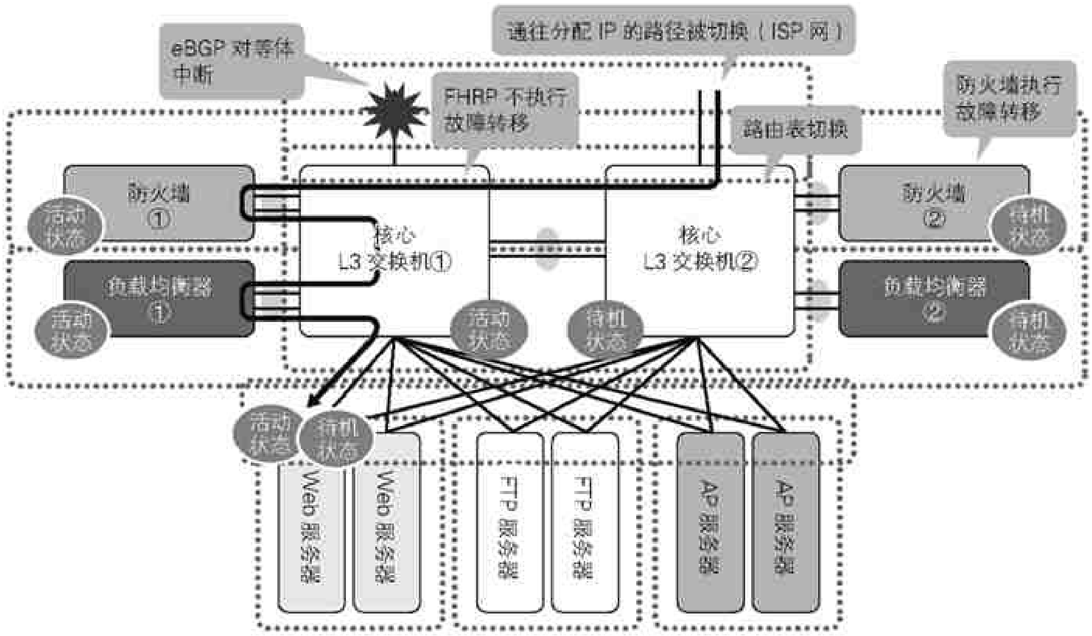


图 **4.2.19**　线路发生故障时会通过 **BGP** 切换路径

- 防火墙故障

当防火墙发生故障时，只有防火墙会立即执行故障转移，其他部分则维持原状，不会执行故障转移。前面已经提到过，在串联式结构中会由 L2 交换机生成一条迂回路径，在单路并联式结构中这一角色则由核心交换机来担任，因此通信会多次经过核心交换机，然后才最终抵达服务器。

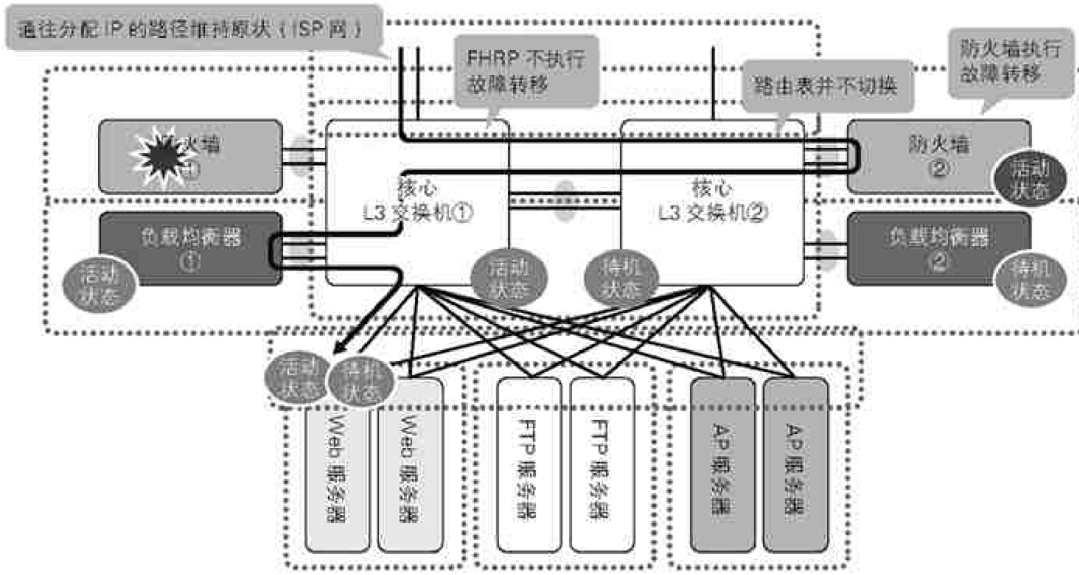


图 **4.2.20**　防火墙发生故障时，只有防火墙会执行故障转移

- 负载均衡器故障

当负载均衡器发生故障时，备用负载均衡器会检测到该故障并立即执行故障转移，其他部分则维持原状，不会执行故障转移。这时候通信会依次经过核心交换机①、防火墙①、核心交换机①、核心交换机②，新升级的活动负载均衡器②、核心交换机②和核心交换机①，像这样多次经过核心交换机之后，才最终抵达服务器。

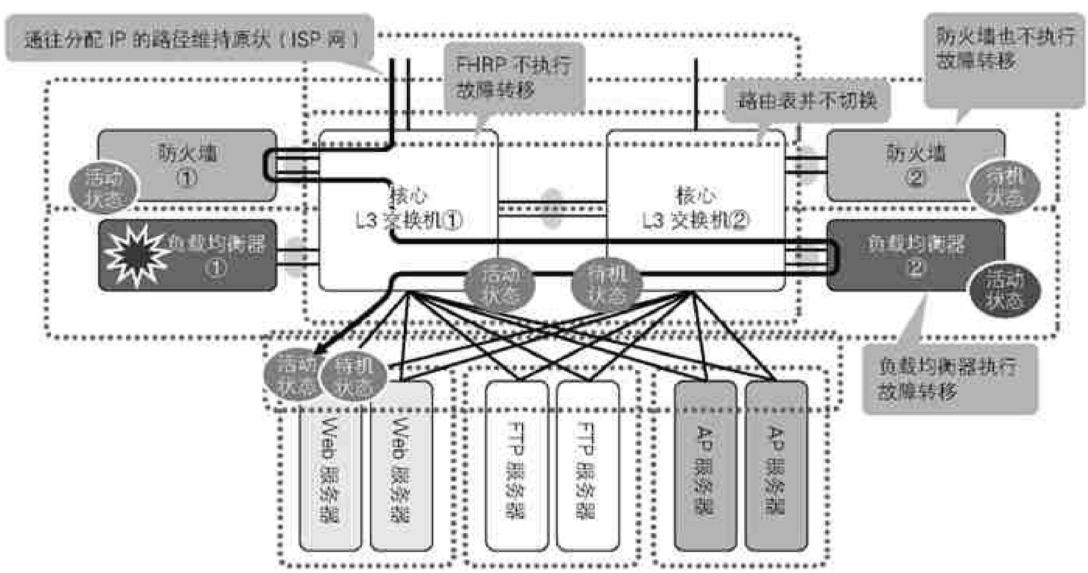


图 **4.2.21**　负载均衡器发生故障时，只有负载均衡器会执行故障转移

- 核心交换机故障

在单路并联式结构中，核心交换机故障的影响范围是非常之大的，在它的影响下所有环节都会执行故障转移。下面我们就来逐一了解一下。

首先，我们来看核心交换机。在外部，由于 eBGP 对等体中断，核心交换机会进行包括 ISP 网在内的路径重新计算；在内部，由于备用机收不到 FHRP 报文，核心交换机会执行故障转移。

接下来，我们再看防火墙和负载均衡器。原本是活动防火墙的防火墙①受到孤立，防火墙②则升级为新的活动防火墙。负载均衡器方面也会发生同样的改变，原本是活动负载均衡器的负载均衡器①受到孤立，负载均衡器②则升级为新的活动负载均衡器。

最后，我们来看服务器网卡。服务器链路因核心交换机的故障而中断，网卡组合检测到该情况之后便会执行故障转移。

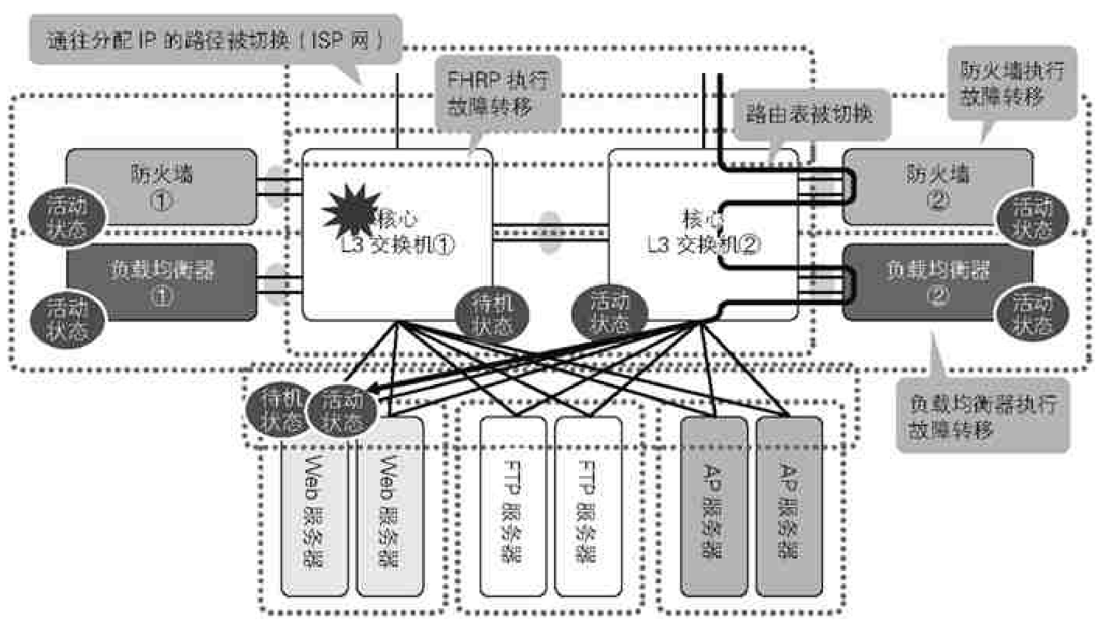


图 **4.2.22**　负载均衡器发生故障时，所有环节都会执行故障转

移

- 服务器的链路故障和网卡故障

  当服务器的链路或网卡发生故障时，只有网卡组合会执行故障转移。这时候，通信会依次经过核心交换机①、防火墙①、核心交换机①、负载均衡器①、核心交换机①和核心交换机②，像这样多次经过核心交换机之后，才最终抵达服务器。

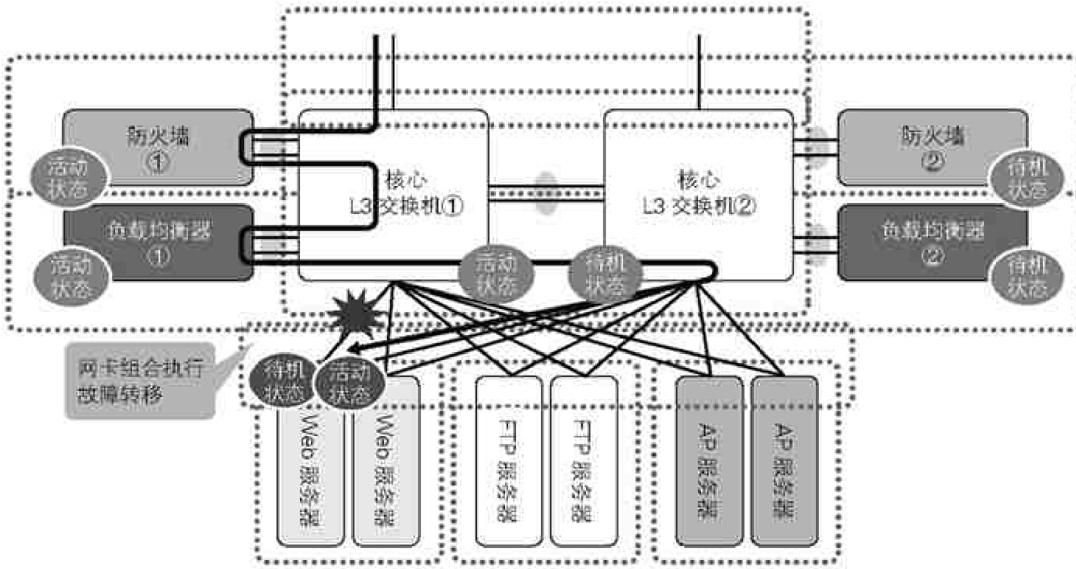


图 **4.2.23**　服务器的链路或网卡发生故障时，只有网卡组合会执行故障转移

- 服务器的服务故障

  这里和单路并联式结构一样，当服务器的服务发生故障时，负载均衡器的健康检查会失败，于是该服务器被排除到负载均衡的对象之外，这样连接就不会被分配到已发生服务故障的服务器上。其他部分没有任何变化。

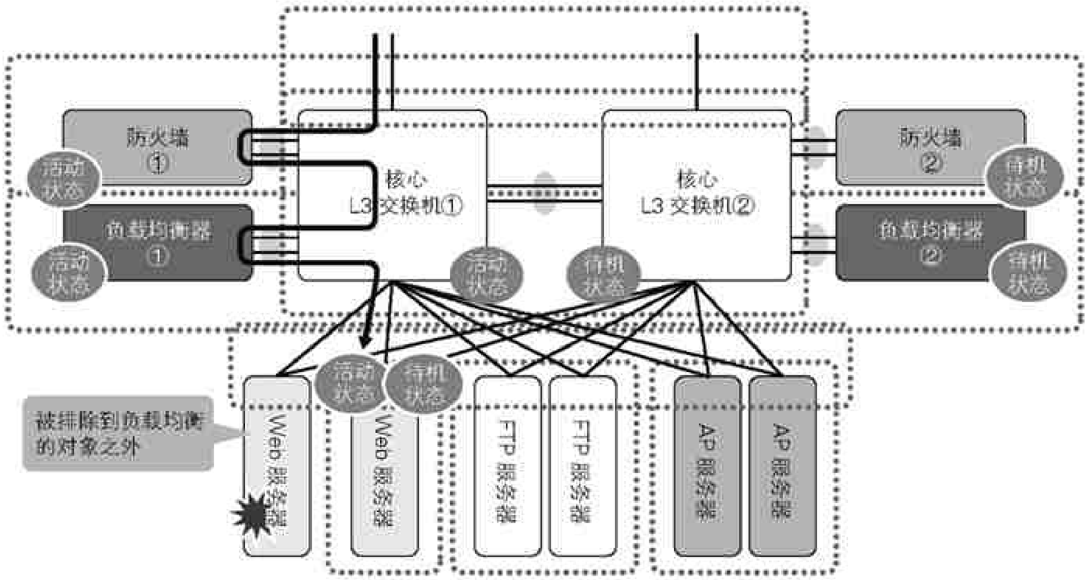


图 **4.2.24**　服务器的服务发生故障时，该服务器会被排除到负载均衡的对象之外

- 非服务器链路的链路故障

服务器链路是由一条条链路构成的，所以发生故障时必须通过网卡组合去执行故障转移。与此相对，其他链路（如核心交换机、防火墙、负载均衡器这些连接网络设备的链路）发生故障时，由于它们都通过链路聚合做了冗余备份，所以无论哪个环节发生了故障都不会执行故障转移，路径也不会被切换。

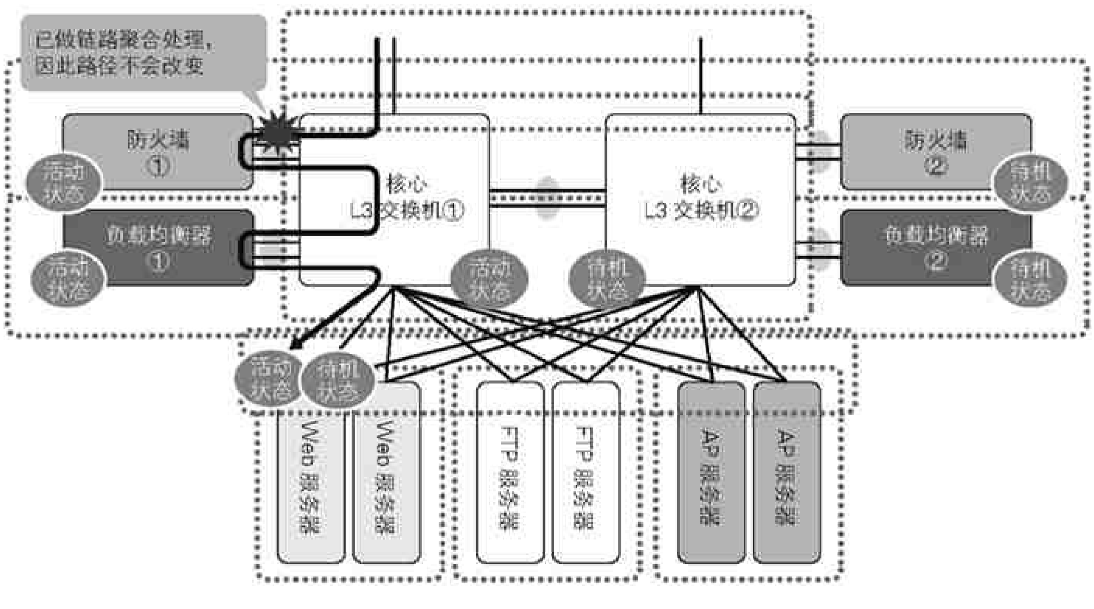


图 **4.2.25**　非服务器的链路由于做了链路聚合处理，路径不会改变

- 非活动路径的故障

  如果是非活动路径发生故障，那么故障无论是发生在什么环节的都不会执行故障转移，也不存在通信中断时间。不过从笔者的经验来说，偶尔也会有这样的情况：因某些网络设备本身的设计漏洞等原因导致其发出了不必要的 GARP，结果离奇宕机。为了避免类似情况的发生，我们应提前做好充分的故障测试，确保各个设备都不会受到不良影响。

第 5 章　管理设计

本章概要

本章将要讲解在服务器端运行管理中使用的技术及其设计要点，以及对于运行管理，我们应该提前规定好的一些事项。

关键任务的服务器端经常会发生形形色色的问题。设计和架构结束之后，服务器端才算是真正开始执行它的使命。在长期使用的过程中，设备可能会发生故障，线缆也可能会断掉。对于这些不同种类的问题，我们不仅要注重发现和预防，还要能够在突发情况下迅速采取对策。为此，我们必须熟练掌握相关技术和设备规格，设计出最符合实际情况的运行管理环境。

5.1　管理技术

管理技术能够让网络更加顺畅地运行，同时提供相应的规则帮助我们管理好网络。网络设计和架构结束之后，一切才刚刚开始，对于将来可能发生的问题，我们只有运用多种管理技术才能快速高效地把它们解决掉。下面，本书将从大量的管理技术中选出比较常用的协议，以及设计时必须注意的事项来逐一讲解。

5.1.1　用 **NTP** 同步时间

NTP（Network Time Protocol，网络时间协议）是一种用于同步设备时间的协议。乍一听可能有人会想“什么？同步设备时间没什么意义吧？”曾几何时，笔者也是这么认为的。但后来，当笔者面对突发问题冷汗涟涟的时候，才终于理解了这个协议是多么重要。

对于牵涉到多台设备的复杂问题，解决的关键在于我们对该问题的理解程度。具体来说，就是能够理解该问题随着时间的变化是如何演变的，即在不同的时刻（几点几分几秒）都发生了什么。为了理清这些头绪，我们需要正确掌握时间这一要素。

NTP 的工作原理非常简单

NTP 的工作原理非常简单。形象一点讲，就是先由客户端发出一个“现在几点了？”的询问（NTP Query），然后服务器针对该询问给出答复（NTP Reply）——“现在是○○点○○分○○秒哦”。

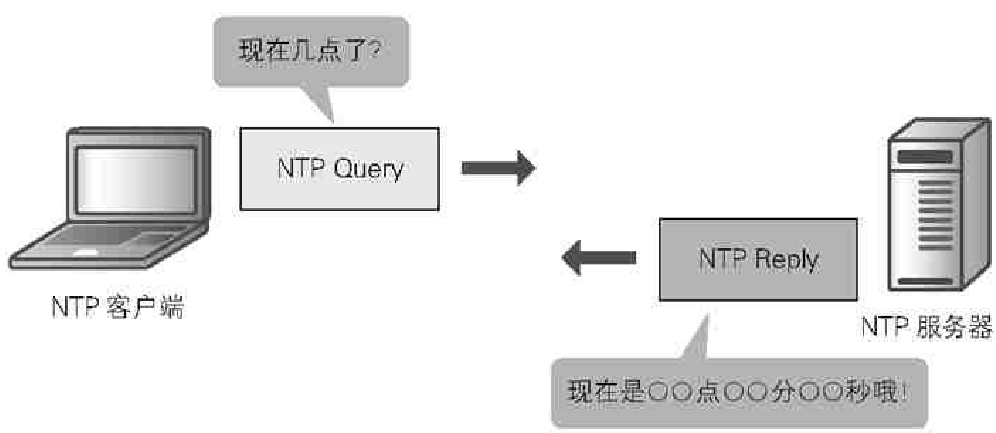


图 **5.1.1　NTP** 的工作原理非常简单

NTP 利用一个叫作“层”（Stratum）的概念及其数值形成阶梯式结构。层值表示从顶层的时间源到本层有多少 NTP 跳。顶层的时间源是从原子钟或 GPS 时钟等获取高精度的时间信息，能够保证时间的正确性，层值为 0。从顶层往下，每经过一台 NTP 服务器，层值就会加一。层值不为 0 的 NTP 服务器对与上一层的 NTP 服务器来说是 NTP 客户端，对下一层的 NTP 客户端来说则是 NTP 服务器。此外，如果无法和上一层的 NTP 服务器同步时间，就绝不会将时间发布给下一层。

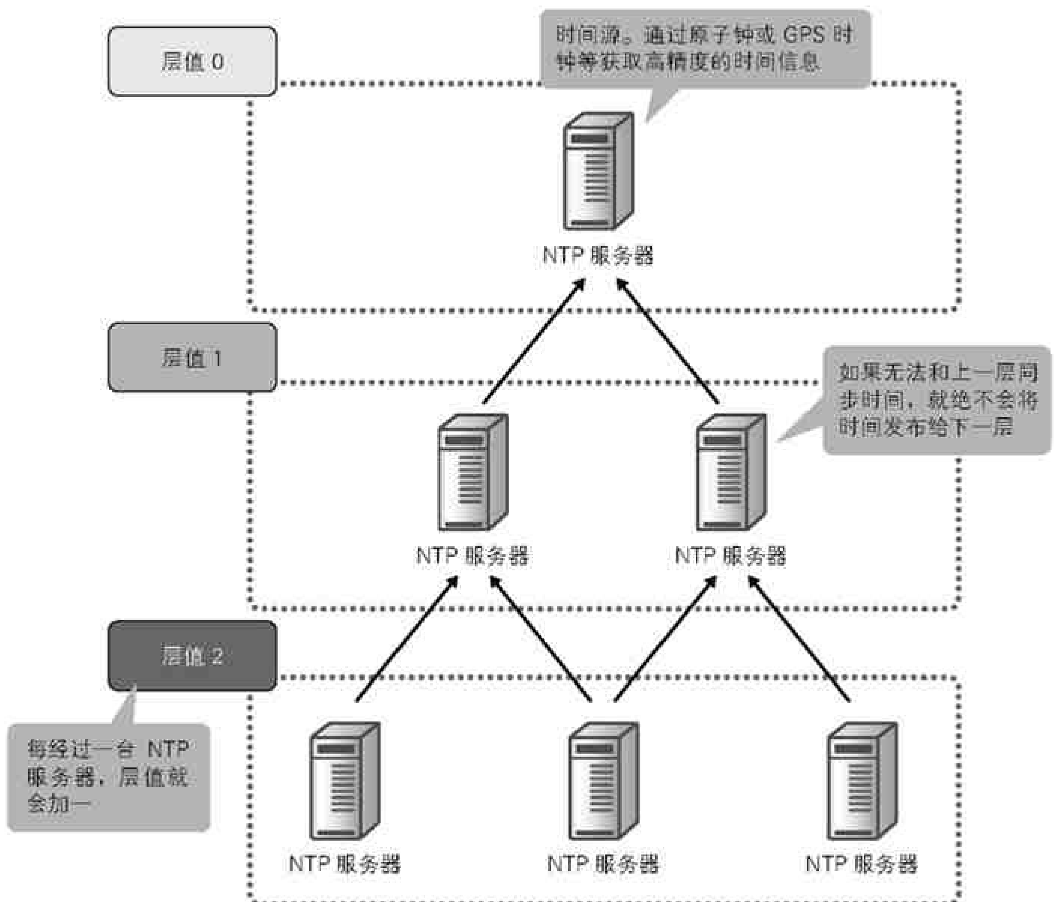


图 **5.1.2　NTP** 是通过**“层”**形成的阶梯式结构

使用 **UDP** 单播

NTP 在任何通信类型中都能够运作，包括单播、多播和广播，不过用于服务器系统的 NTP 仅限单播。本书将仅介绍单播的 NTP。

- **NTP Query**

  前面已经讲过，“现在几点了？”这个询问就是 NTP Query。在 NTP Query 报文中，源 IP 地址为需要同步时间的设备的 IP 地址，目的 IP 地址则是 NTP 服务器的 IP 地址。采用的协议为追求实时性的 UDP。此外，源端口号和目的端口号都是 123。通过标志字段中的模式数值可以识别该报文是 Query 还是 Reply。如果是 Query 则数值为 3（client）。

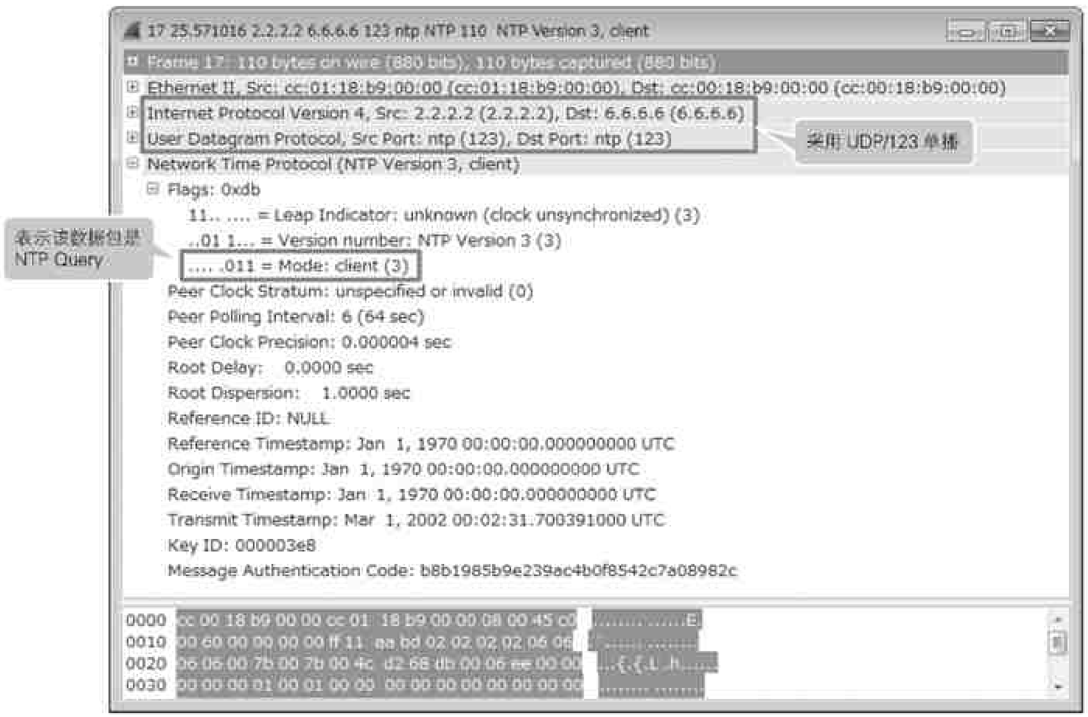


图 **5.1.3**　通过 **UDP** 单播发送 **NTP Query**

- **NTP Reply**

  前面已经讲过，“现在是○○点○○分○○秒哦”这个答复就是 NTP Reply。在 NTP Reply 报文中，源 IP 地址为 NTP 服务器的 IP 地址，目的 IP 地址则是需要同步时间的设备的 IP 地址。和 NTP Query 一样，采用的协议也是追求实时性的 UDP，源端口号和目的端口号也都是 123。通过标志字段中的模式数值可识别该报文是 Query 还是 Reply。如果是 Reply 则数值为 4（server）。模式后面紧跟着客户端生成正确时间所需要的信息，包括层、时间信息和延迟，等等。

同步的时间间隔会发生改变

使用的 NTP 应用程序不同，对 NTP 服务器执行同步处理的时间间隔就不同。我们以 Linux 中常用的 ntpd 为例来看一下。ntpd 的同步时间间隔在尚不稳定的时候为 64 秒，随着稳定程度的提高会逐渐变成 128 秒、256 秒……像这样成倍地增加，最长可达 1024 秒。当然，这些时间间隔的设置是可以修改的。

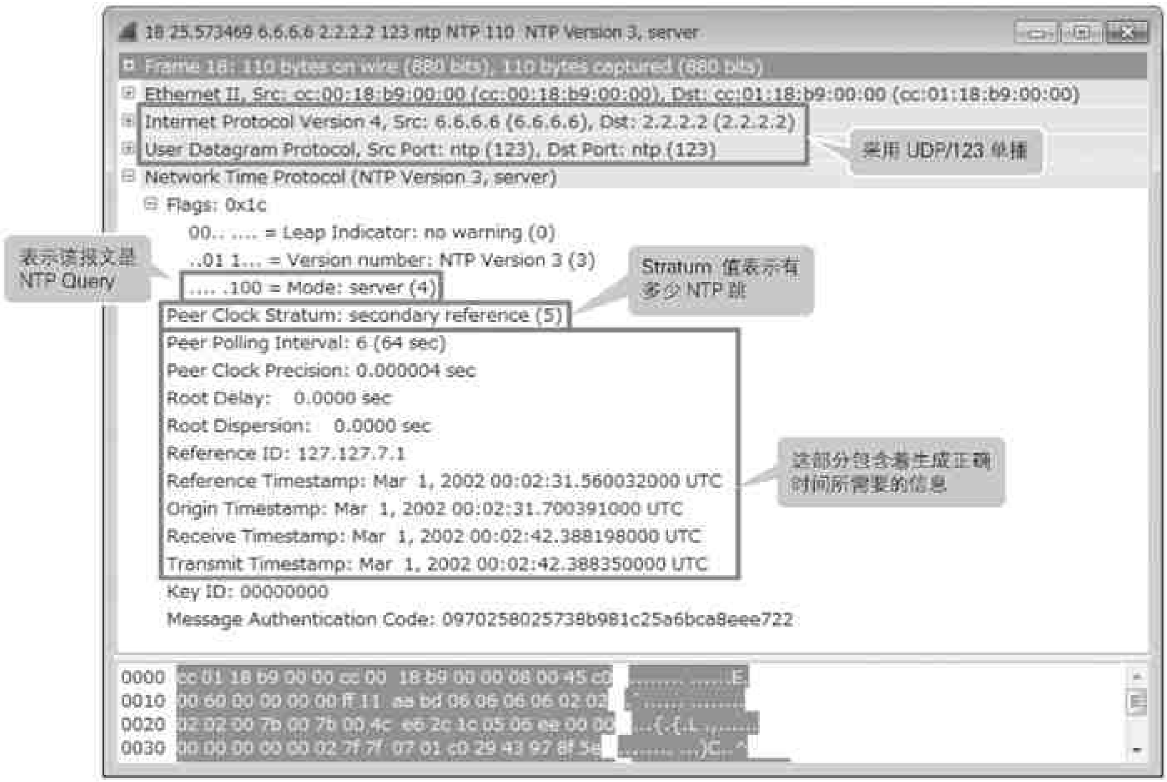


图 **5.1.4　NTP Reply** 中包含着诸多生成正确时间所需要的信息

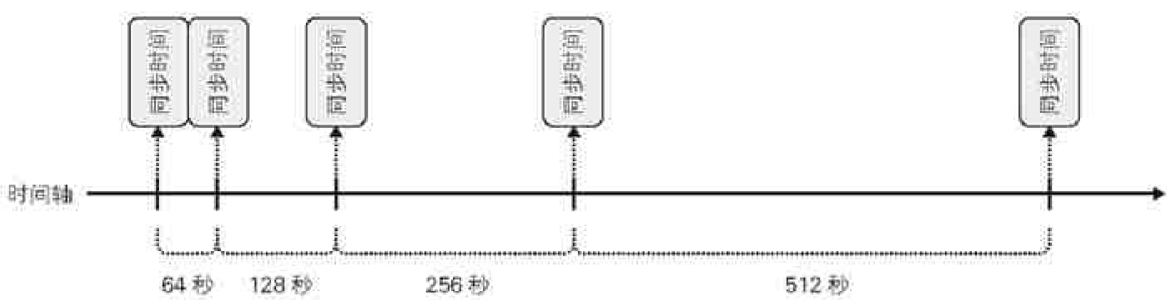


图 **5.1.5**　同步的时间间隔会逐渐变长

两种运行模式

NTP 客户端有两种运行模式，分别叫作 step 模式和 slew 模式。粗略地讲，二者的区别在于它们对服务器发布的时间是如何校准的。step 模式的校准是一气呵成地完成的，slew 模式的校准则是慢条斯理地实现的。采用哪一种模式取决于 NTP 客户端使用的 OS。例如，ntpd 默认的运行模式就是当 NTP 客户端的时间和 NTP 服务器发布的时间相差超过 128 毫秒时采用 step 模式，小于 128 毫秒时则采用 slew 模式。

- **step** 模式

  在 step 模式中，无论 NTP 客户端的时间和 NTP 服务器发布的时间相差多少，校准的工作都是一气呵成的。就算 NTP 客户端的计时又往前走了一点，也会被调回成 NTP 服务器发布的时间。交换机、路由器等并不特别注重时间信息的设备一般会采用这种模式。

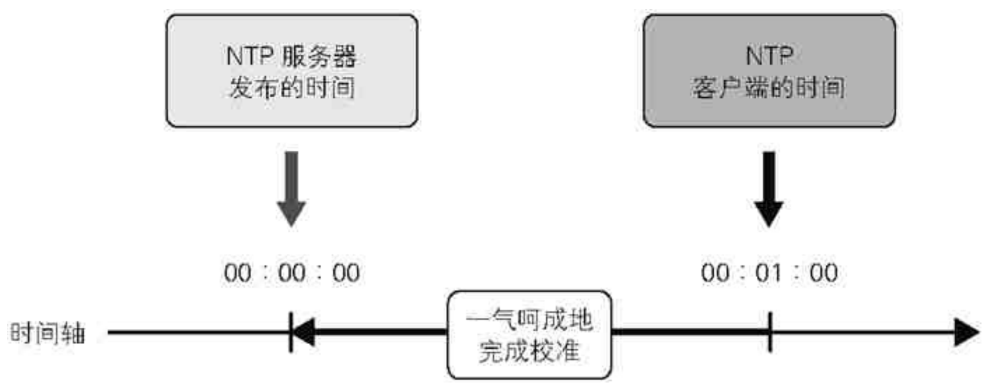


图 **5.1.6**　**step** 模式一气呵成地完成调对

- **slew** 模式

  在 slew 模式中，校准是慢条斯理地完成的，每秒钟只校正 0.5 毫秒，所以就算 NTP 客户端的计时又往前走了一点，也不会被马上调回到 NTP 服务器发布的时间。由于校正的时间比计时往前走的时间要短，所以校准的进度非常缓慢。

  在 DB 应用程序、日志应用程序等时间信息非常重要的应用程序中，采用 step 模式一气呵成地完成校准可能会引发故障，因此人们一般会在安装应用程序之前先用 step 模式校准，之后如果出现时间差异就改用 slew 模式去校准。

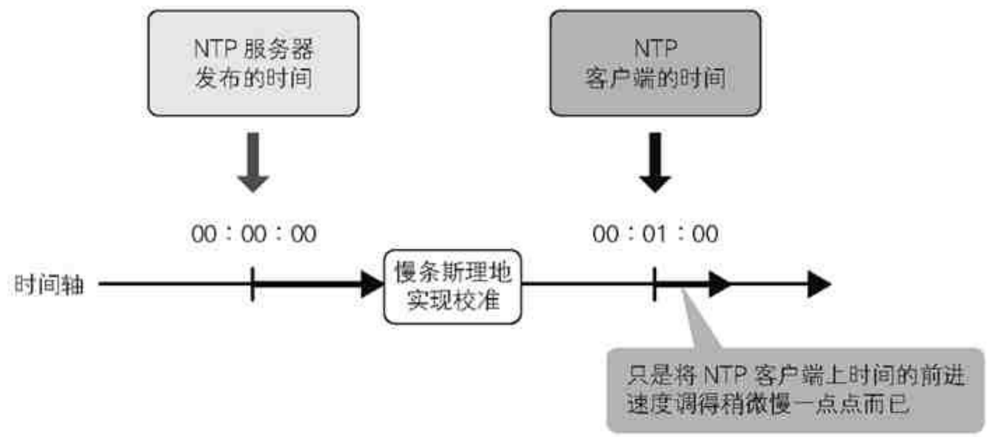


图 **5.1.7**　**slew** 模式慢条斯理地实现校准

要耐心等待时间校准结束

在实际架构服务器系统的过程中，由于各台设备的时间往往五花八门、各不相同，很容易让我们烦躁不安。有的设备有强制发送 NTP Query 的命令，这样还好说。但如果没有类似的命令，那么该设备同步时间的作业就会很费工夫。对于这种情况，我们一定要有足够的耐心才行。不过，如果等了一个小时还是没有校准好，那就说明是设置有误，应该去仔细确认设置的内容。

将监控服务器用作 **NTP** 服务器，同步效果会更好

利用 NTP 同步时间时，最重要的事项就是保持整个系统的统一性。系统内的各台设备都要和分散在互联网上的 NTP 服务器取得时间上的同步，然而，如果这些 NTP 服务器的时间本身

就不统一，那么即使各台设备完成了同步也是毫无意义的。而且，还会浪费互联网的通信流量。所以，我们应在系统内只选择一两台 NTP 服务器，然后执行时间同步作业，这样就能够保证整个系统的统一性，效率也会更高。

那么，我们应该选择哪一台（或两台）服务器，让其担任 NTP 服务器这一角色呢？这是个经常会被问到的问题。当然，我们也可以把 NTP 服务器完全独立出来。不过，如果要让 NTP 服务和其他服务器的服务共存，那么笔者建议让 NTP 服务器和监控服务器共居一体。这是因为，时间信息在监控服务器的日志中最能发挥作用，最能成为我们排除故障的重要依据。只要校准监控服务器的时间，

我们就能将日志按照时间顺序有条不紊地整理出来。

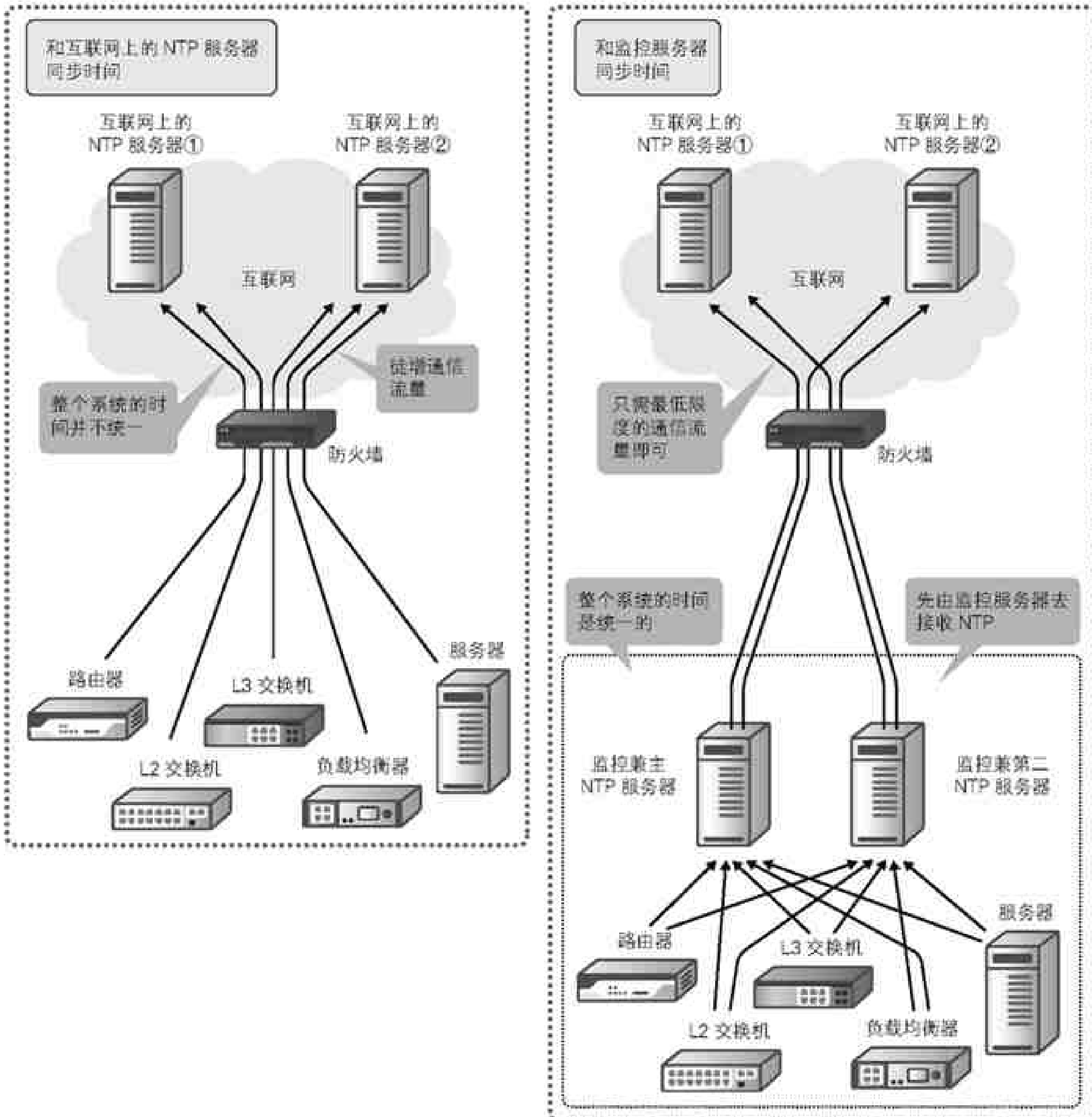


图 **5.1.8**　确保系统内时间的一致性

将 **Windows** 服务器用作 **NTP** 服务器时的注意点

如果是将 Windows 服务器用作 NTP 服务器，我们需要注意一点。Windows 服务器中包含一个叫作 W32Time 的时间同步服务，但是在默认状态下，这项服务并不会对其他设备发布时间信息。要想使其发布时间信息，就必须修改它的注册表项，详情情况请参考以下网站。

URL http://support.microsoft.com/kb/816042/

5.1.2 用 SNMP 检测故障

SNMP（Simple Network Management Protocol，简单网络管理协议）是一种用于监控网络设备、服务器的性能以及监控故障的业内标准管理协议，在服务器系统的运行管理中使用得非常广泛。在服务器系统中，任何故障出现之前都是会有蛛丝马迹的，抓住这些线索极其重要。所以，我们应坚持定期地收集所管设备的各种信息，包括 CPU 使用率、内存占用率、通信流量、数据包数量等，尽早检测出故障的苗头。

5.1.2.1 通过 SNMP 管理器和 SNMP 代理交换信息

SNMP 由两个要素构成，分别是身为管理方的 SNMP 管理器和被管理的 SNMP 代理。在这二者之间交换着好几种消息组合，前者通过这些消息组合就能够掌握后者当前的状态。

SNMP 管理器是负责收集 SNMP 代理的管理信息并对其进行监控的应用程序。比较知名的有 MRTG（Multi Router Traffic Grapher，多路由器流量图示器）、Open View Network Node Manager（OpenView 网络节点管理器）和 TWSNMP 管理器[1]等。无论哪一种应用程序都能对收集来的信息进行加工并基于 Web GUI 实现可视化，让用户对信息一目了然。

> [1] 由 Twise Labo. 公司的山居正幸先生开发的一款 SNMP 管理器。详细情况可参考 http://www.twise.co.jp/twsnmp.html 。——译者注

SNMP 代理则是接受 SNMP 管理器发出的要求并将故障通知给对方的程序。绝大部分的网络设备和服务器中都装有 SNMP 代理。在 SNMP 代理中有一个叫作 MIB（Management Information Base，管理信息库）的树状数据库，该数据库中保存着通过 OID（Object Identifier，对象标识符）数值识别出来的管理信息以及与这些信息相关的数值。SNMP 管理器发来的要求中就包括 OID 值，SNMP 代理者根据该值返回相关值，而当发现 OID 值有变时，就会通知 SNMP 管理器发生了故障。

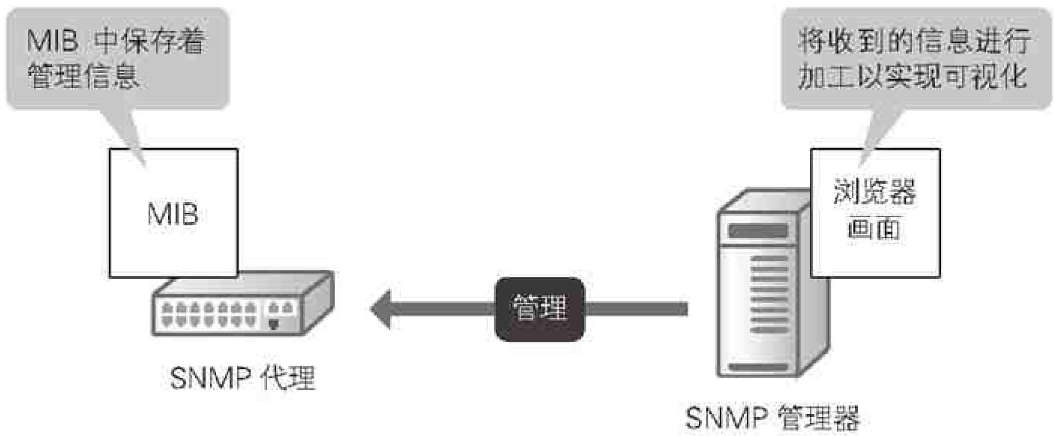


图 **5.1.9**　**SNMP** 管理器对 **SNMP** 代理进行管理

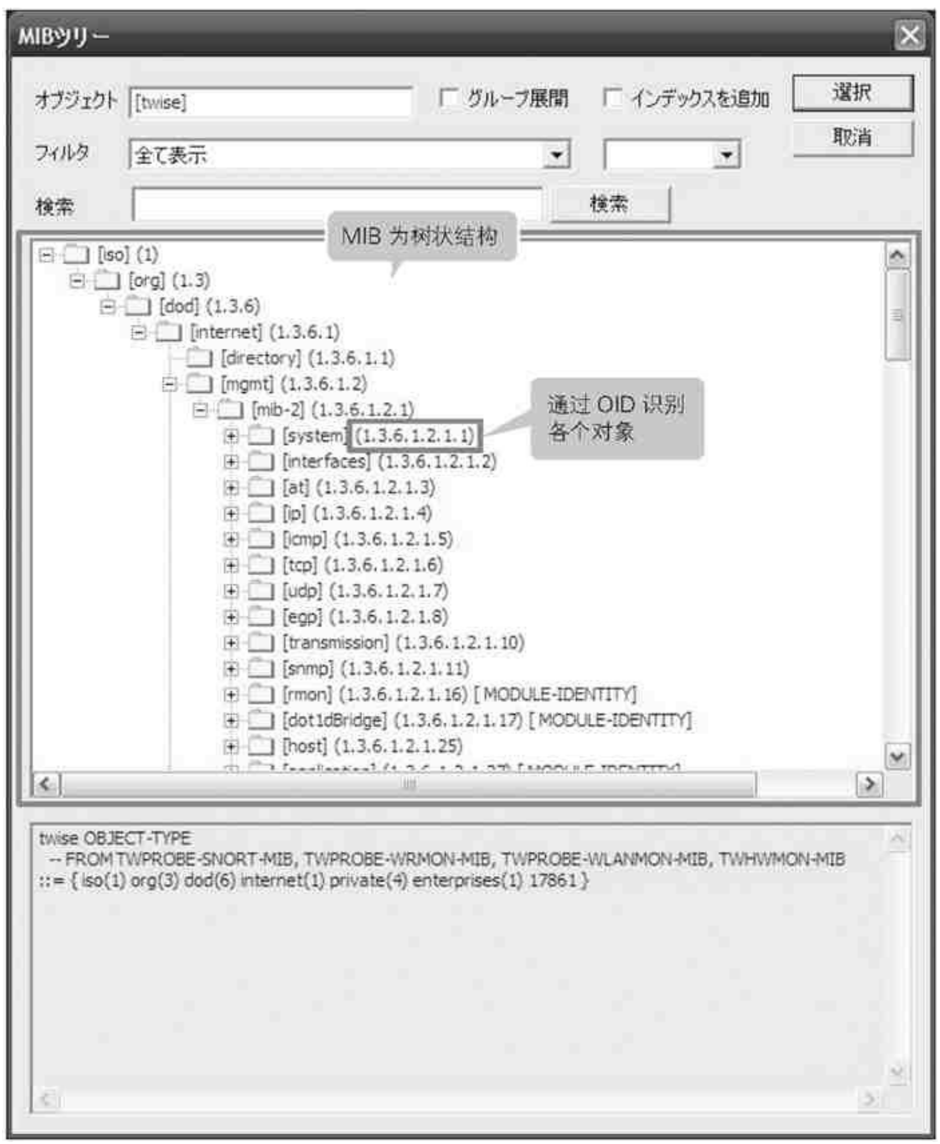


图 **5.1.10**　**MIB** 为树状结构（图为在 **TWSNMP** 管理器中看到的 **MIB** 信息）

5.1.2.2　熟练掌握三种运作模式

SNMP 采用 UDP，工作原理非常简单易懂。将 GetRequest、GetNextRequest、SetRequest、GetResponse 和 Trap 这五种消息进行组合，然后通过 SNMP Get、SNMP Set 和 SNMP Trap 这三种运作模式实现沟通。各消息之间均以 Community 名为认证口令，口令通过则通信成立。下面我们来分别看看这三种类型是如何运作的。

- **SNMP Get**

  SNMP Get 是获取设备信息的运作模式。形象地说就相当于 SNMP 管理器提出“请把○○的信息给我！”的请求，SNMP 代理则给出“这个信息的内容是○○哦！”的回应，过程并不复杂。

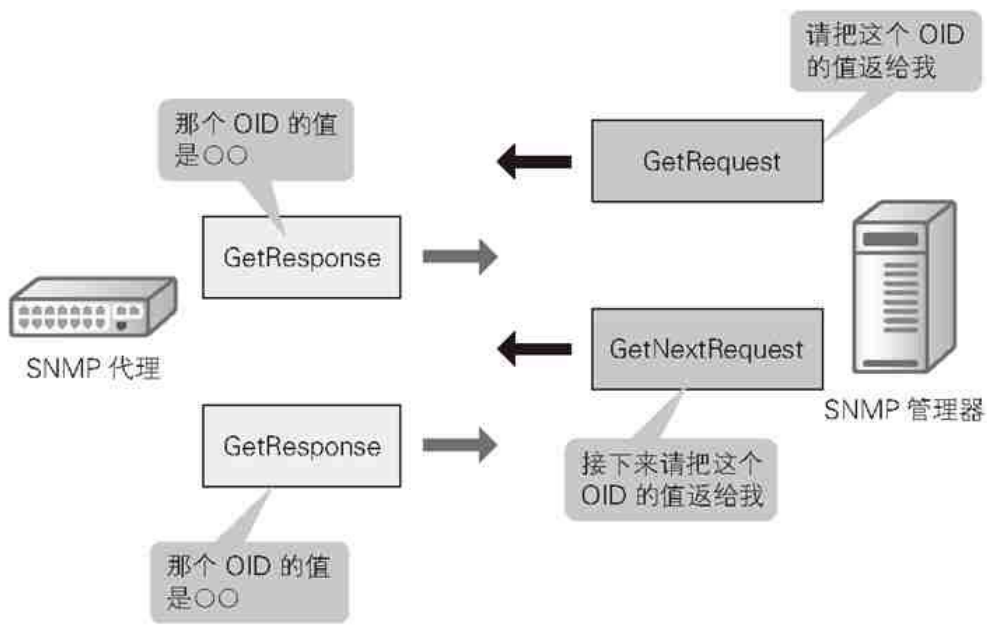


图 **5.1.11**　通过 **SNMP Get** 获取 **OID** 的信息

具体的步骤就是先由 SNMP 管理器将包含着 OID 的 GetRequest 消息发给 SNMP 代理。GetRequest 消息以 UDP 单播的形式发出，目的端口号为 161。然后，SNMP 代理将被指定的 OID 的值以 GetResponse 消息的形式返回给 SNMP 管理器。SNMP 管理器需要下一则信息时，就将所需的 OID 放在

GetNextRequest 消息中再次发给 SNMP 代理，然后 SNMP 代理者又返回相应的 GetResponse 消息。像这样不断循环往复。

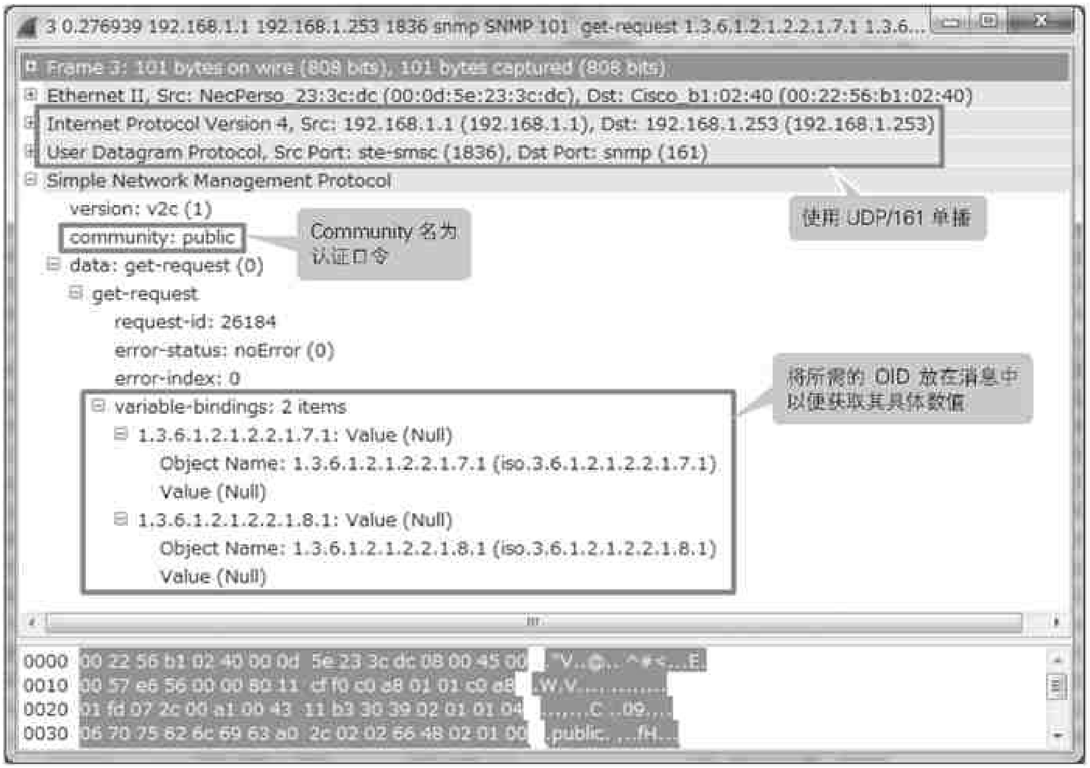


图 **5.1.12**　发出一条包含 **OID** 的 **GetRequest** 消息

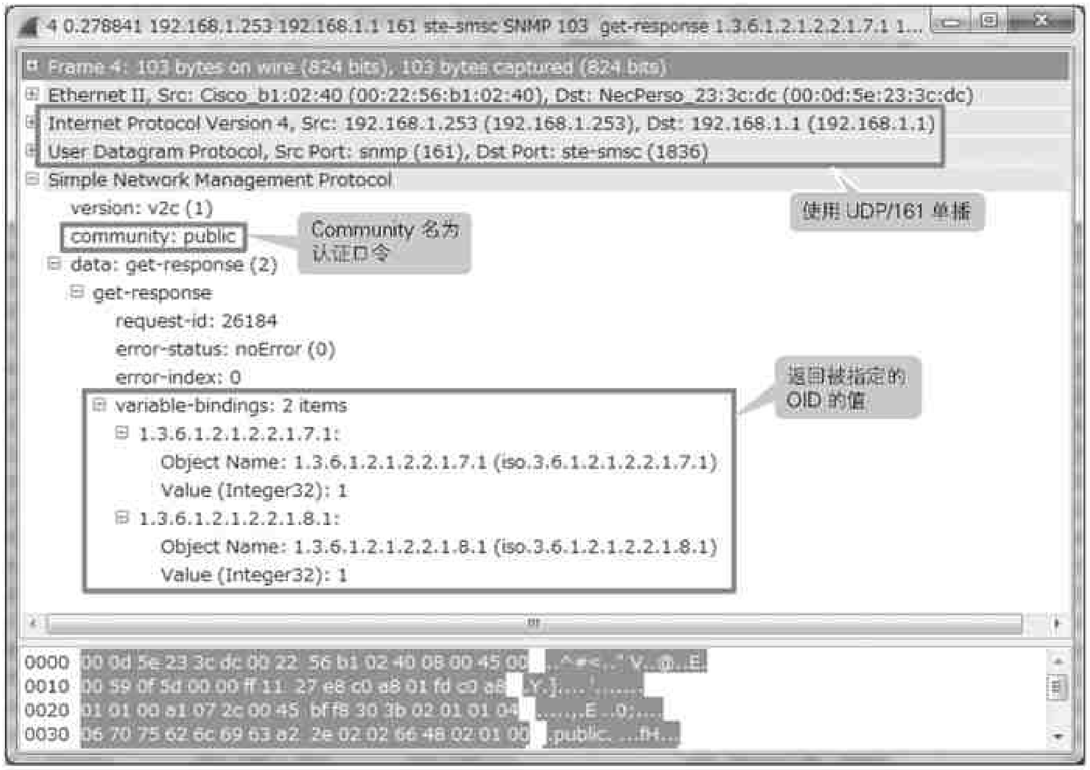


图 **5.1.13**　返回的 **GetResponse** 消息包含了被指定的 **OID** 的值

SNMP Set

SNMP Set 是更新设备信息的运作模式。形象地说相当于 SNMP 管理器提出了“请更新○○的信息！”的要求，SNMP 代理执行之后则给出“更新好了！”的回应。以前曾有系统管理人员问笔者“为什么要更新呢？那不成了造假吗？”但实际上，这种运作并不是用来造假的。在使用 SNMP Set 的例子当中，最容易理解的就是关闭端口了。我们知道，SNMP 代理保存的 OID 值代表了端口的状态，只有更新这个值之后我们才能将端口关闭掉。

具体步骤和 SNMP Get 的差别并不大，只是使用的消息不同而已。先是 SNMP 管理器将包含着 OID 的 SetRequest 消息发给 SNMP 代理。SetRequest 消息和 GetRequest 消息同样以 UDP 单播的形式发出，目的端口号也是 161。然后，SNMP 代理将更新好的 OID 值以 GetResponse 消息的形式返回给 SNMP 管理器。

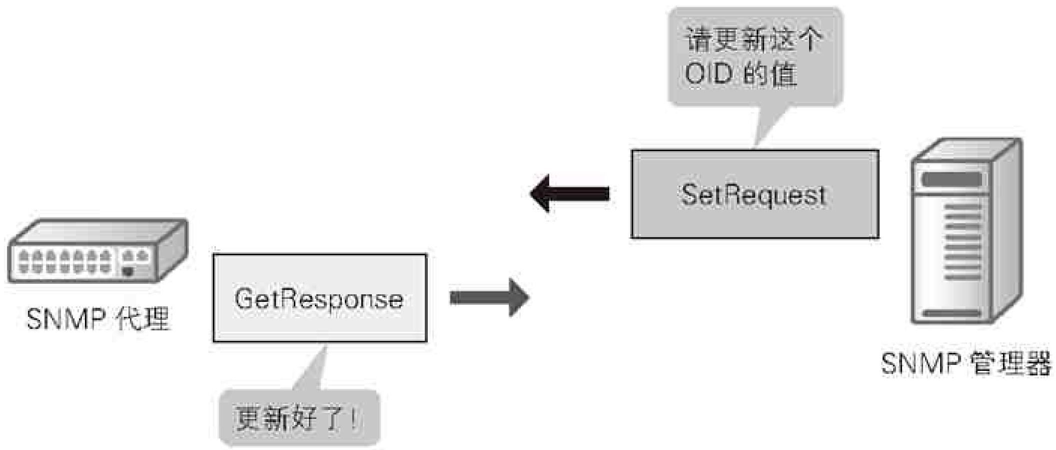


图 **5.1.14**　通过 **SNMP Set** 更新 **OID** 的值



图 **5.1.15**　发出 **SetRequest** 消息，告知需要更新的 **OID** 及其当前数值

SNMP Trap

SNMP Trap 是告知故障发生的运作模式。形象地说就相当于 SNMP 代理向 SNMP 管理器发出了“○○发生故障了！”的通知。这里请务必要注意，SNMP Get 和 SNMP Set 都是由 SNMP 管理器发出的通

信，只有 Trap 是由 SNMP 代理发出的通信。

SNMP 代理一旦发现 OID 的值有了特定的变化，就会断定有故障发生，于是向 SNMP 管理器发送 Trap 信息。Trap 依然是以 UDP 单播的形式发出，但目的端口号是 162。

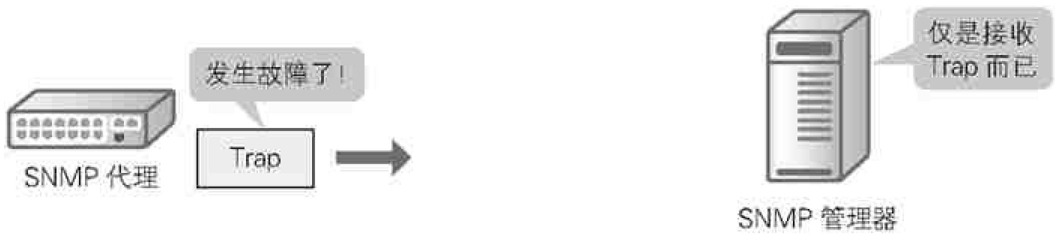


图 **5.1.16**　通过 **SNMP Trap** 检测到故障

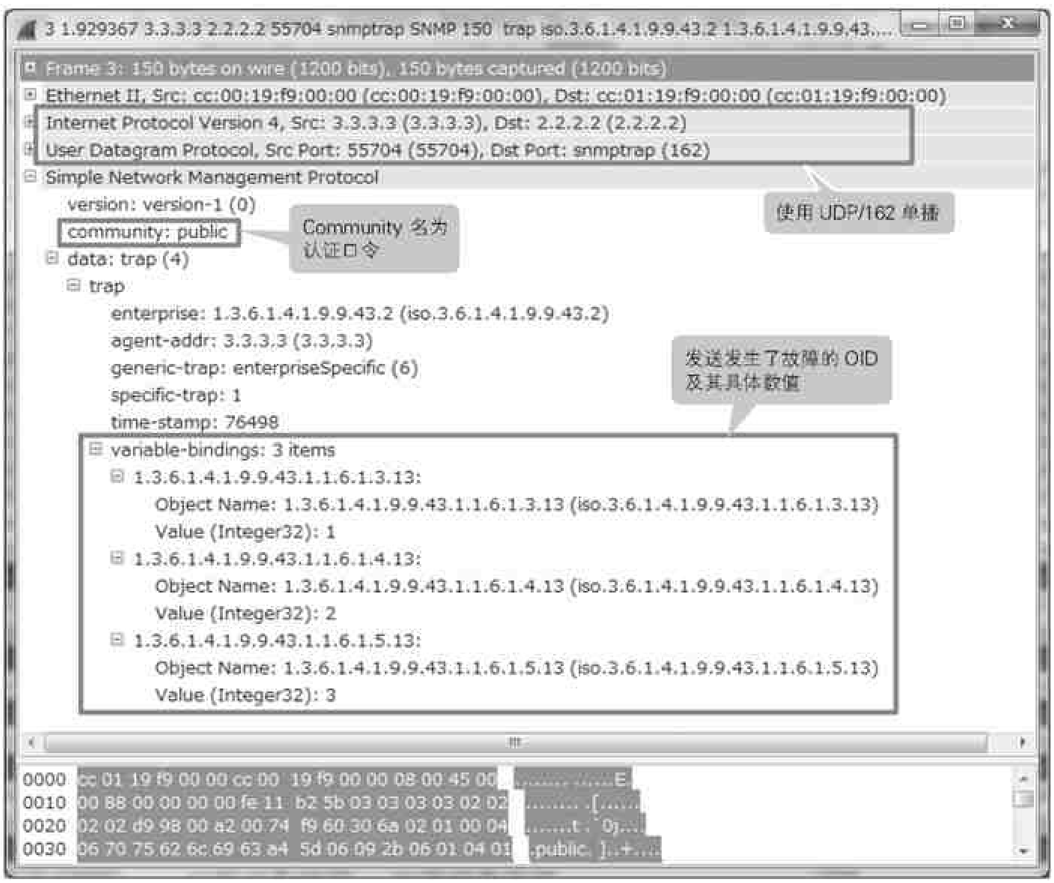


图 **5.1.17**　通过 **SNMP Trap** 告知是哪个 **OID** 发生了故障

5.1.2.3　限制源 IP 地址

目前，世界上最常用的 SNMP 版本为 v2c 版本。v2c 并不具备加密功能，重要的管理信息会以明文的形式发出。在 v2c 中，唯一能够保证数据安全的只有 community 名这个认证口令，但信息终究还是以明文的形式发出去的，所以数据安全是该版本的一个弱项。

为了解决这个问题，我们在使用 SNMP 的时候务必要限制被 SNMP 代理所允许的源 IP 地址、即 SNMP 管理器的 IP 地址，以此来保证数据安全。

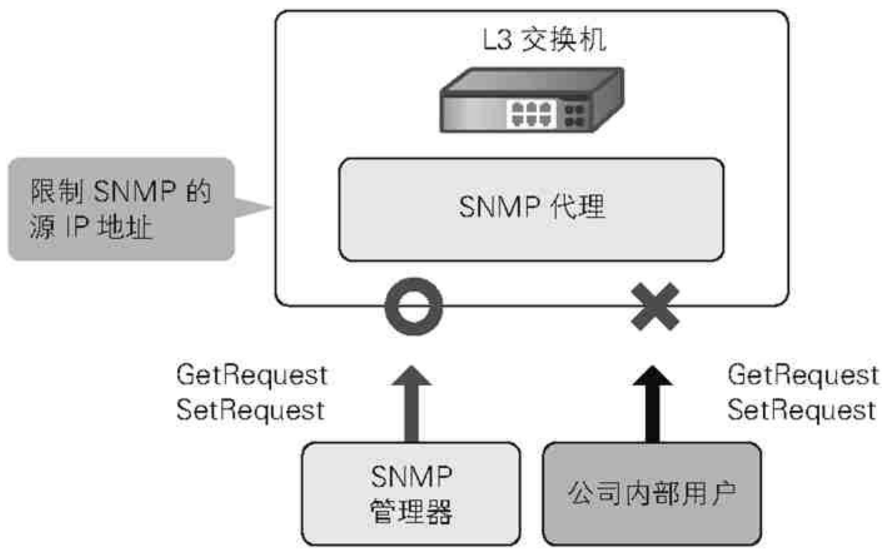


图 **5.1.18**　在 **SNMP** 代理中限制源 **IP** 地址

5.1.3　用 Syslog 检测故障

Syslog 是一种用于转发日志消息的业内标准管理协议。网络设备和服务器的内部（缓存、硬盘等）保存着很多记录了各种事件的日志，Syslog 将这些日志转发给 Syslog 服务器，实现日志的一元化管理。在服务器系统中，Syslog 常常和 SNMP 一起使用。

Syslog 的工作原理非常简单

Syslog 采用 UDP，工作原理非常简单易懂。当发生某个事件的时候，它会将该事件保存到本机的缓冲或硬盘中，同时也将事件转发给 Syslog 服务器。转发以 UDP 单播的形式进行，目的端口号为 514。消息部分由 Facility（程序模块）、Severity（严重性）和消息本体构成，消息本体即日志的内容，本书将要介绍的是 Facility 和

Severity。



图 **5.1.19**　通过 **Syslog** 转发日志

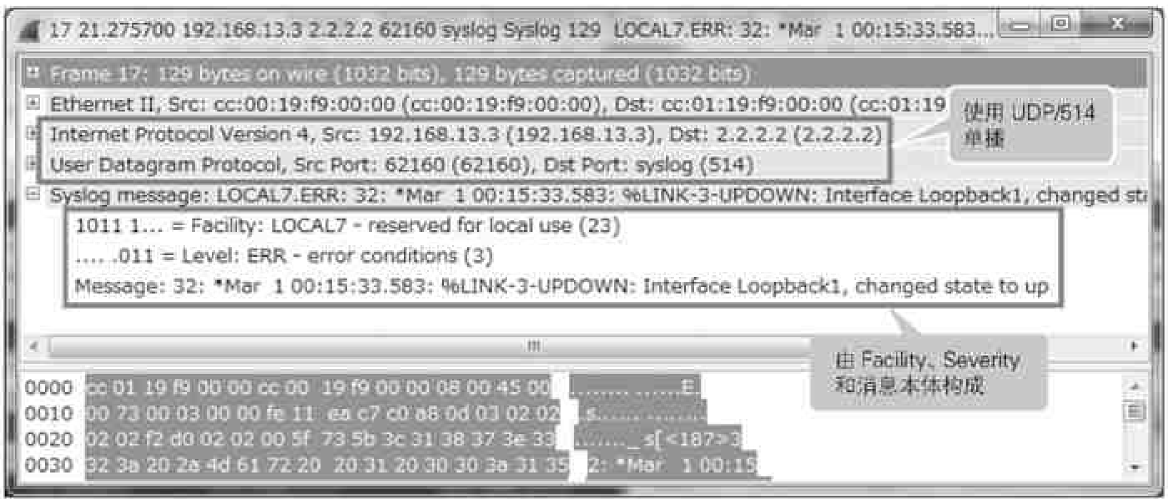


图 **5.1.20**　通过 **Syslog** 转发日志消息

Severity

Severity 是表示日志消息重要性的值，分成 0 ～ 7 共八个等级，值越小，重要性就越高。哪个 Severity 以上的消息应发给 Syslog 服务器、哪个 Severity 以上的消息应保存到什么程度（消息大小或保存时间），这些事项在设计的时候都必须定义清楚。例如，“warning 级别以上的消息应发给 Syslog 服务器，informational 级别以上的消息应保存在缓存中（最大 40960 字节）”——应该像这样去定义。对有些设备来说，输出日志这个作业本身可能就会成为处理负荷，所以我们也要充分考虑到这一点，合理调整 Severity，避免让输出日志成为处理负荷。

表 **5.1.1**　**Severity** 的值表示重要性的高低

名称	解说	Severity	重要性
emergencies	使系统不稳定的错误	0	高
alerts	必须紧急处理的错误	1	↑
critical	致命的错误	2	
errors	错误	3	
warnings	警告	4	
notifications	通知	5	
informational	参考信息	6	↓
debugging	调试	7	低

Facility

Facility 表示日志消息的种类，总共有 24 种，具体内容如下表所示。

表 **5.1.2　Facility** 表示消息的种类

Facility	代码	解说
kern	0	内核消息
user	1	任意用户的消息
mail	2	邮件系统（sendmail、qmail等）的消息
daemon	3	系统守护进程（ftpd、named等）的消息
auth	4	数据安全/认证（login、su等）的消息
syslog	5	Syslog守护进程的消息
lpr	6	行式打印机子系统的消息

news	7	网络新闻子系统的消息
uucp	8	UUCP子系统的消息
cron	9	时钟守护进程（cron和at）的消息
auth-priv	10	数据安全/认证的消息
ftp	11	FTP守护进程的消息
ntp	12	NTP子系统的消息
—	13	日志审计的消息
—	14	日志警告的消息
—	15	时钟守护进程的消息
local0	16	任意用途
local1	17	任意用途
local2	18	任意用途
local3	19	任意用途
local4	20	任意用途
local5	21	任意用途

local6	22	任意用途
local7	23	任意用途

请注意，有些设备的 Facility 是不能修改的，所以设计时我们一定要仔细确认好设备的规格。

将消息过滤后会更加容易管理

笔者常听说这样的情况：在 Syslog 服务器的日常运行当中，日志消息总是在系统发生故障的时候大量涌现，结果导致管理人员看漏了重要的日志。其实，故障发生时收到大量的日志消息是再自然不过的事情，而这种关键时刻如果起不到任何作用，Syslog 服务器也就毫无存在意义了。为了避免看漏重要的日志，我们应通过 Severity 或 Facility 在 Syslog 服务器中对消息进行过滤。

5.1.4　传递设备信息

服务器系统发生故障的根源大多在物理层，所以我们必须做好连接管理。具体说来，就是要清楚哪个端口连接着哪台设备，这在服务器系统的运行管理中极其重要。网络中有好几个能够查找相邻设备的协议，都是我们管理连接的好帮手，比较有代表性的是 CDP（Cisco Descovery Protocol，思科发现协议）和 LLDP（Link Layer Discovery Protocol，链路层发现协议）。二者都是 L2 协议，能够发送包括 IP 地址、机型和 OS 版本等在内的设备信息，让连接管理变得轻松起来。

5.1.4.1　CDP

CDP 是思科公司独有的 L2 协议。对设备信息进行封装处理时，它使用的不是 Ethernet Ⅱ（DIX）规格，而是 IEEE802.3 规格的增强版 IEEE802.3 with LLC/SNAP。

CDP 为激活状态的设备在默认情况下会每隔 60 秒将一个 CDP 帧发送给系统已占用的多播 MAC 地址 01-00-0c-cc-cc-cc。收到 CDP 帧的互连设备将其保存到缓存中，在接下来的 180 秒之内如果没有收

到对方发来的 CDP 帧，或者链路发生了中断，该设备就会将对方的相关链路信息丢弃。

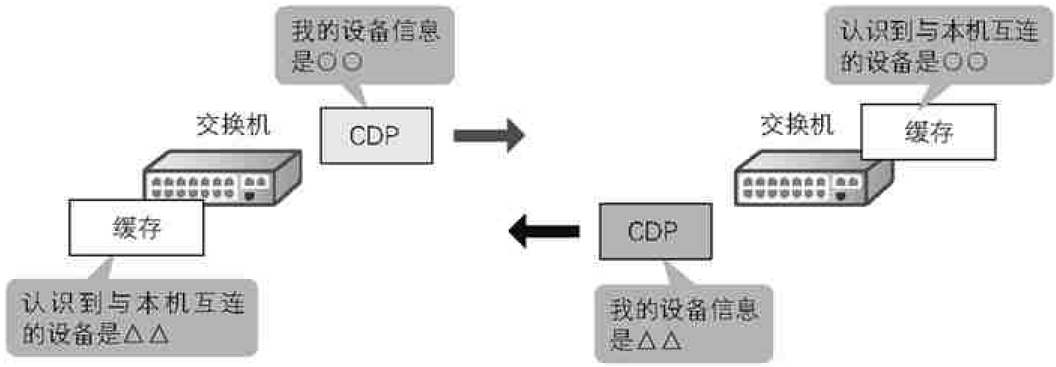


图 **5.1.21**　通过 **CDP** 了解相邻设备的信息

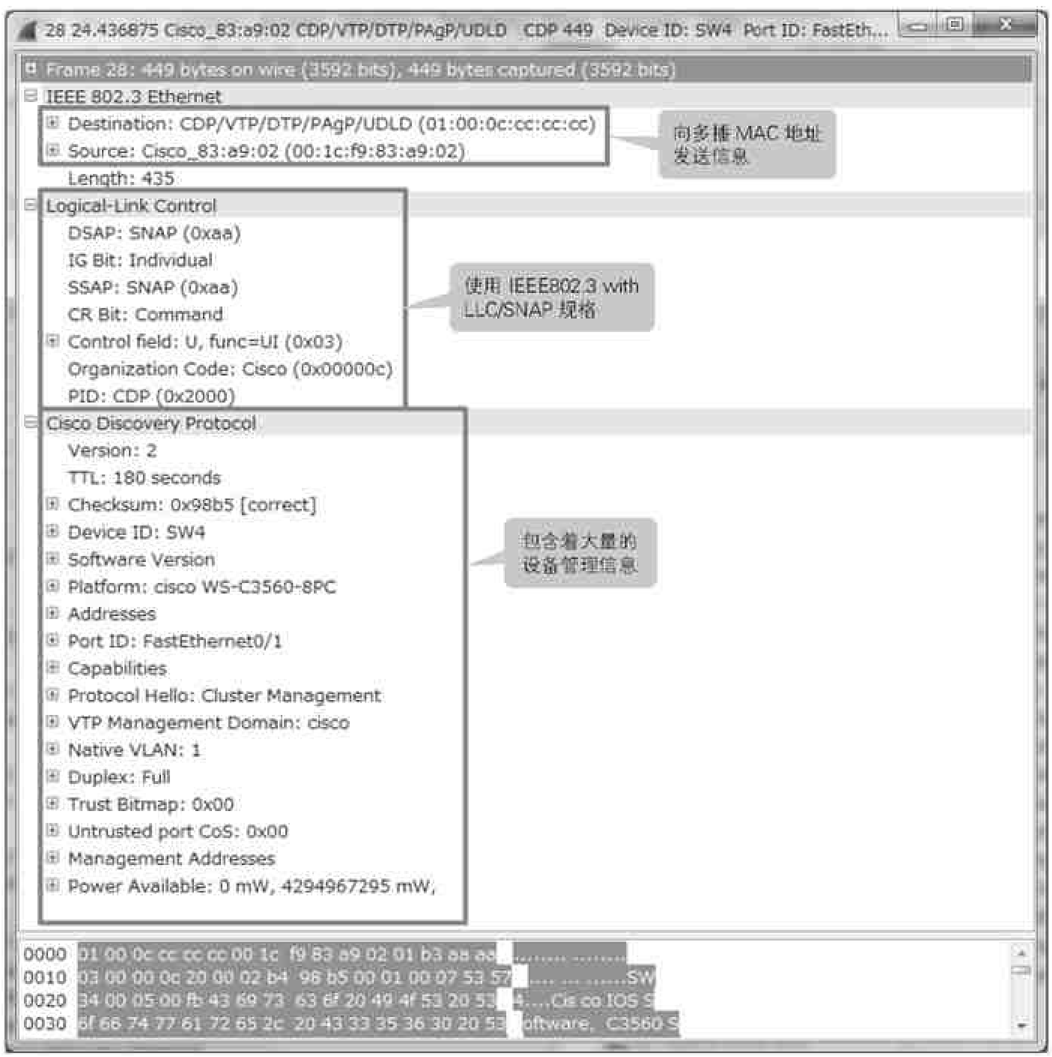


图 **5.1.22**　**CDP** 中包含着诸多的设备管理信息

5.1.4.2　LLDP

LLDP 是基于 IEEE802.1ab 的协议，一般用在由多家供应商提供的不同品牌设备所组成的环境中。它使用 Ethernet Ⅱ规格对设备信息进行封装处理，格式和 CDP 有所不同，但功能基本上是一样的。

LLDP 为激活状态的设备每隔 30 秒（推荐秒数）将一个 LLDP 帧发送给系统已占用的多播 MAC 地址 01-80-c2-00-00-0e。收到 LLDP 帧的互连设备将其保存到一个叫作 LLDP MIB 的数据库中管理。在接下来的 120 秒之内如果没有收到对方发来的 LLDP 帧或者链路发生了中断，该设备就会将对方的相关链路信息丢弃。

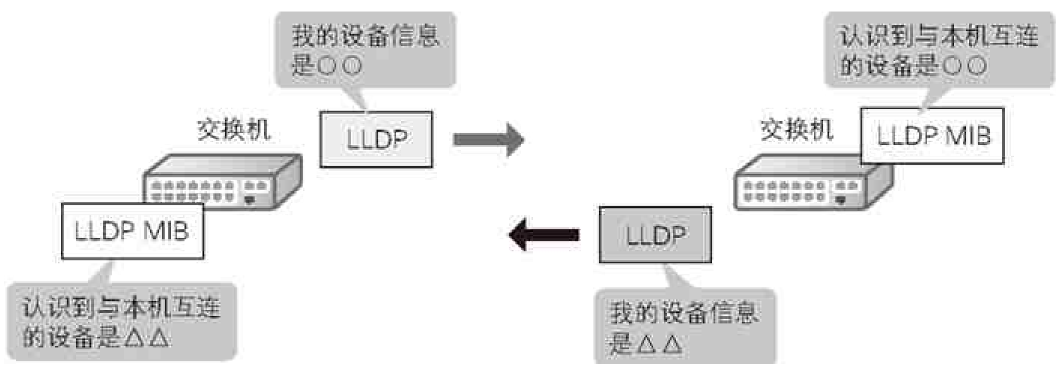


图 **5.1.23**　通过 **LLDP** 了解相邻设备的信息

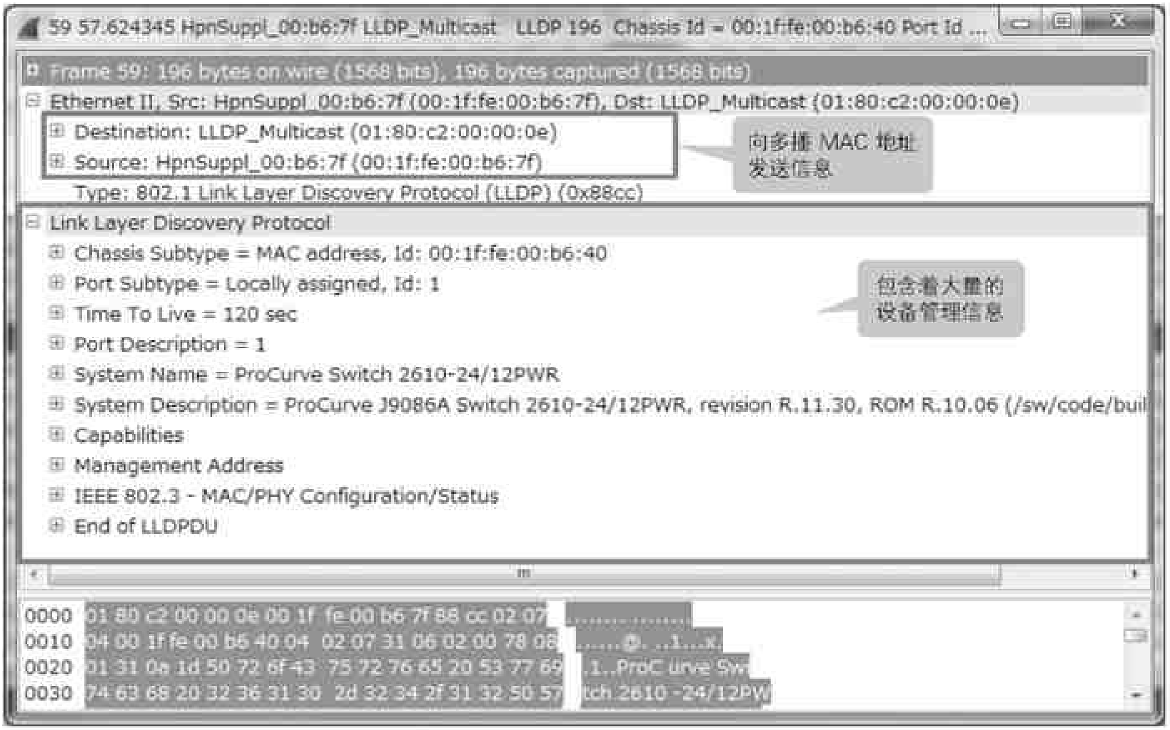


图 **5.1.24　LLDP** 中包含着诸多的设备管理信息

5.1.4.3　注意 **CDP** 和 **LLDP** 的数据安全问题

CDP 和 LLDP 在连接管理方面能够发挥巨大的作用，然而它们并不是完美无缺的。由于二者都是以明文的形式发送设备管理信息，从数据安全的角度来看极易受到恶意攻击。因此，在 Untrust 区段和 DMZ 区段，我们最好不要激活这两个协议。笔者一般是这样会在连接设备的时候激活这两个协议，等到连接结束并且确认到物理结构的连接没有问题之后就将它们注销，用这种方式来应用 CDP 和 LLDP。

总之，进行设计的时候，我们首先应该决定是否需要激活查找相邻设备的协议。如果需要，那么接下来就要明确两个事项：应激活哪一个协议以及在哪里激活（用在何处）。间隔时间（发送信息的间隔）和 Hold 时间（经过多长时间之后丢弃信息）一般无需变更。

5.2　管理设计

从本节开始，将不再讲述运行管理技术，而是介绍一些偏管理方面的设计事项。内容会稍微琐碎一点，但对今后的工作是大有裨益的，建议大家对这些内容在设计阶段就一一做好规定。

5.2.1　确定主机名

主机名定义了了地点、设备以及设备承担的角色等标识符，必须让人一看就懂才行。有些设备规定主机名中不能使用特殊字符或者必须使用 FQDN，对于这些限制，我们一定要预先确认好。在定义主机名时有两个要点容易被忽视，那就是位数和称呼。

在大规模的网络环境中，人们常常会利用 Tera Term 或 Excel 的宏处理等自动形成主机名。主机名的位数如果一致，宏的制作就会相对简单一些，可以省去很多不必要的作业。此外，称呼也是一个容易被人们忽视的要点。主机名中包含着各种各样的标识符，理解不够充分的话很容易看得晕头转向。所以，我们应定义和主机名一致的称呼，这样在谈论相关话题或编写文档等情况下就不会弄混了，可谓好处多多。

5.2.2　通过标签管理连接

所谓定义标签，就是规定在什么地方贴什么样的标签，这在管理设计中是个非常重要的环节。你可能会想“啊？标签？封条？能起多大的作用啊？”千万别小看一枚小小的标签，它的作用可大着呢。系统发生故障的时候，我们需要立即知道是什么地方的哪台设备出了问题，而这个小小的标签就能帮上我们的大忙[2]。

> [2] 出于对数据安全的考虑，有些网络环境可能不允许使用标签。遇到这种情况时应和系统管理人员协商解决。

5.2.2.1　线缆标签

这里说的线缆不光是 LAN 线缆，还包括电源线缆、用于 SAN 的光纤光缆等服务器端的所有线缆。对于这些线缆，我们要定义它们应

各自使用哪一种标签以及要标注什么内容。

定义的办法有很多种，如果在现有的其他系统中也用到了标签，我们不妨直接取经，以该系统中的定义为准。一般来说，人们大都会使用过塑类型的标签或圆形标签，在标签上注明连接源设备和连接目的设备的主机名和端口号。

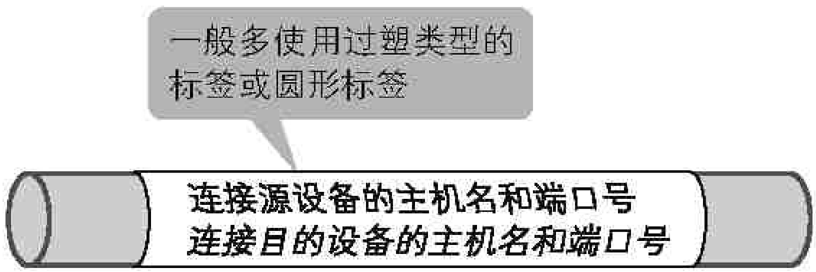


图 **5.2.1**　定义线缆标签的种类和标注的内容

5.2.2.2　本体标签

我们对贴在设备本体上的标签也应做出定义，定义将它们贴在哪里以及标注些什么内容。本体标签同样有多种定义的办法，如果在现有的其他系统中也用到了标签，我们不妨直接取经，以该系统中的定义为准。有些管理人员会对本体标签中的字体大小比较介意，我们最好详细征询一下他们的意见。一般来说，设备本体的正、反面都应贴上标有主机名的标签。

5.2.3　设计密码

在管理设计中，我们还要为设备定义其专用密码。无论放置设备的数据中心建得有多么固若金汤，如果我们将设备密码随便地设成“password”，那么数据安全依然无从谈起，因为这种密码毕竟只是虚张声势的纸老虎而已。根据信息安全认证规格ISMS（Information Security Management System，信息安全管理体系）的规定，好的密码必须具备以下几个条件。

（1）容易记忆。

（2）他人很难从本人的相关信息（如名字、电话号码、生日等）中推测或获取该密码。

（3）经得住字典式攻击（即无法从字典中包含的单词或短语破解密码）。

（4）不是重复同样的文字，不仅是数字或字母的长串罗列。

如果并不需要获得 ISMS 认证，我们大可不必严格遵守上面的条件。不过，至少我们应该避免将密码设置成用户能够轻易想到的内容。有些设备可以在浏览层面和设置层面分别设置几个不同的密码，对这样的设备，我们绝不能图省事而都设成完全一样的密码，否则分层管理就形同虚设。所以我们千万不能偷懒，一定要给不同的层面分别设置不同的密码才行。

5.2.4 管理设置信息

应如何备份网络设备的设置信息、又应如何将它们还原，这也是管理设计的一项重要内容。人们往往十分重视服务器的备份和恢复，对网络设备的备份和恢复却没有足够的关注。然而，网络设备的设置信息承载着整个系统的根基和主干，其重要性绝不亚于服务器中的同类部分。因此，对于网络设备的设置信息，我们一定要明确定义应在何时、何处、怎样地去进行备份和恢复。

5.2.4.1 在备份设计中应定义时机、方式和保存地点

进行备份设计是要把握时机、方式和保存地点这三个要点，下面就简单说明一下这三个要点。

在修改设置之前备份

网络设备的备份不同于服务器的备份，人们一般很少会去定期执行。当然，我们可以通过专用的管理工具去定期执行备份。最近，也可以利用网络设备中的 cron[3] 功能定期安排备份了。不过在大多数情况下，人们都是在修改设置之前去备份的。

> [3] cron 是一种能够定期执行 job 的功能。

为不同的设备定义不同的备份方式

设备不同，备份方式就会有所不同。有些设备甚至专门配置了生成

备份文件用的菜单，足见差别之大。所以，我们务必仔细确认需要备份什么信息以及通过什么协议去执行备份，针对不同的设备定义不同的备份方式。

定义将备份文件保存到什么地方

我们还必须定义应将备份文件保存到哪里。如果将每台设备的备份文件都保存到不同的地方就毫无统一性可言了，只会引起管理上的混乱。另外，我们还要注意备份文件名的格式，同样，如果各自使用不同的格式就会杂乱无章，导致我们会连最基本的设备设置时间都无从得知。总而言之，各个方面我们都必须进行统一的规划，保持管理的统一性。

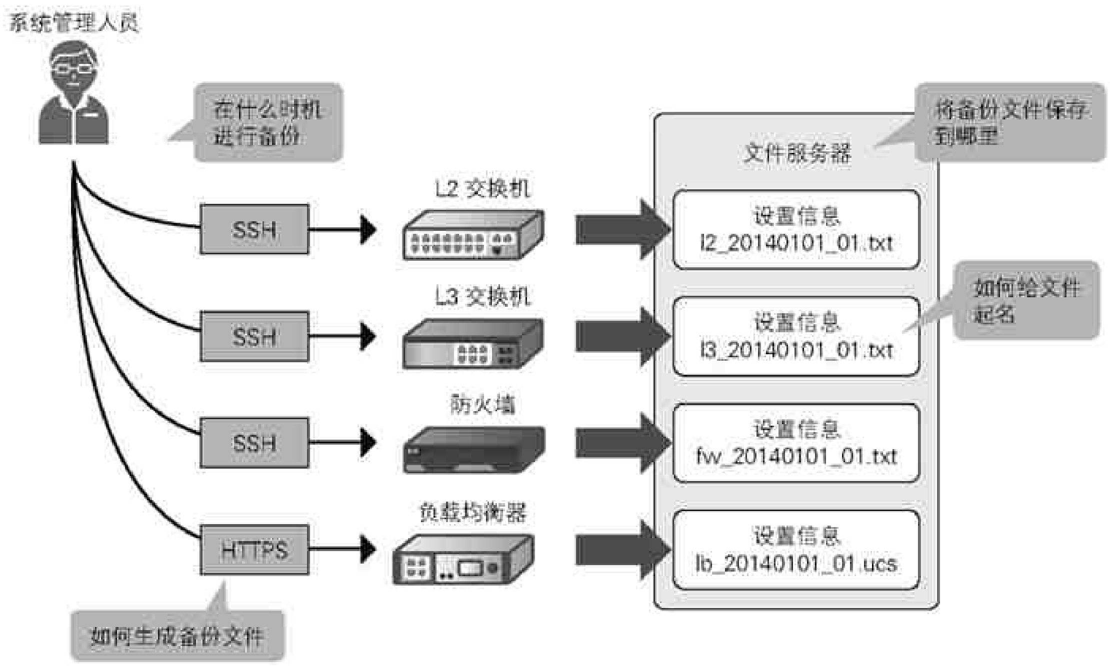


图 **5.2.2**　在备份设计中定义备份的时机、方式和保存地点

5.2.4.2　发生故障时执行恢复处理

和服务器的恢复处理一样，网络设备的恢复处理也是在发生故障时执行的。至于采用什么恢复方式则和备份一样取决于设备本身。我们要预先确认各种设备分别应该采取怎样的恢复方式，然后明确做出定义。对某些设备来说，在发生故障的时候与其执行恢复处理，还不如直接重新设置。对于这样的设备，我们应将其作为例外单独定义。总而言之，在恢复处理的设计中，系统快速起死回生的命门

掌握在清晰明确的各种定义当中。

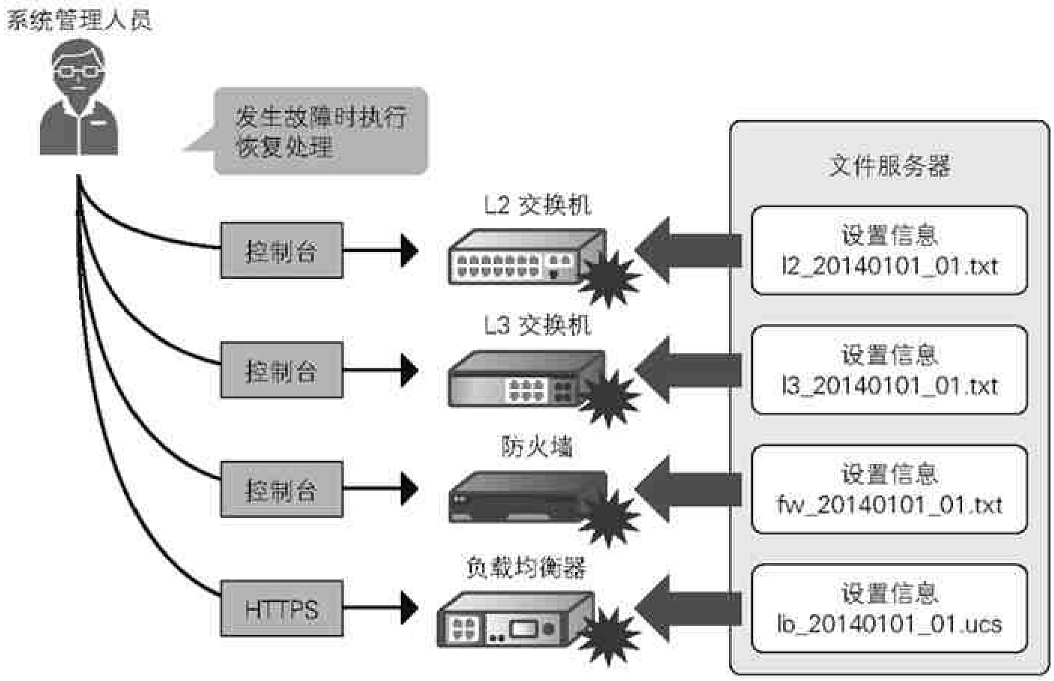


图 **5.2.3**　发生故障时执行恢复处理

看完了

如果您对本书内容有疑问，可发邮件至contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱：ebook@turingbook.com。

在这里可以找到我们：

- 微博 @图灵教育 : 好书、活动每日播报
- 微博 @图灵社区 : 电子书和好文章的消息
- 微博 @图灵新知 : 图灵教育的科普小组
- 微信 图灵访谈 : ituring_interview，讲述码农精彩人生
- 微信 图灵教育 : turingbooks
